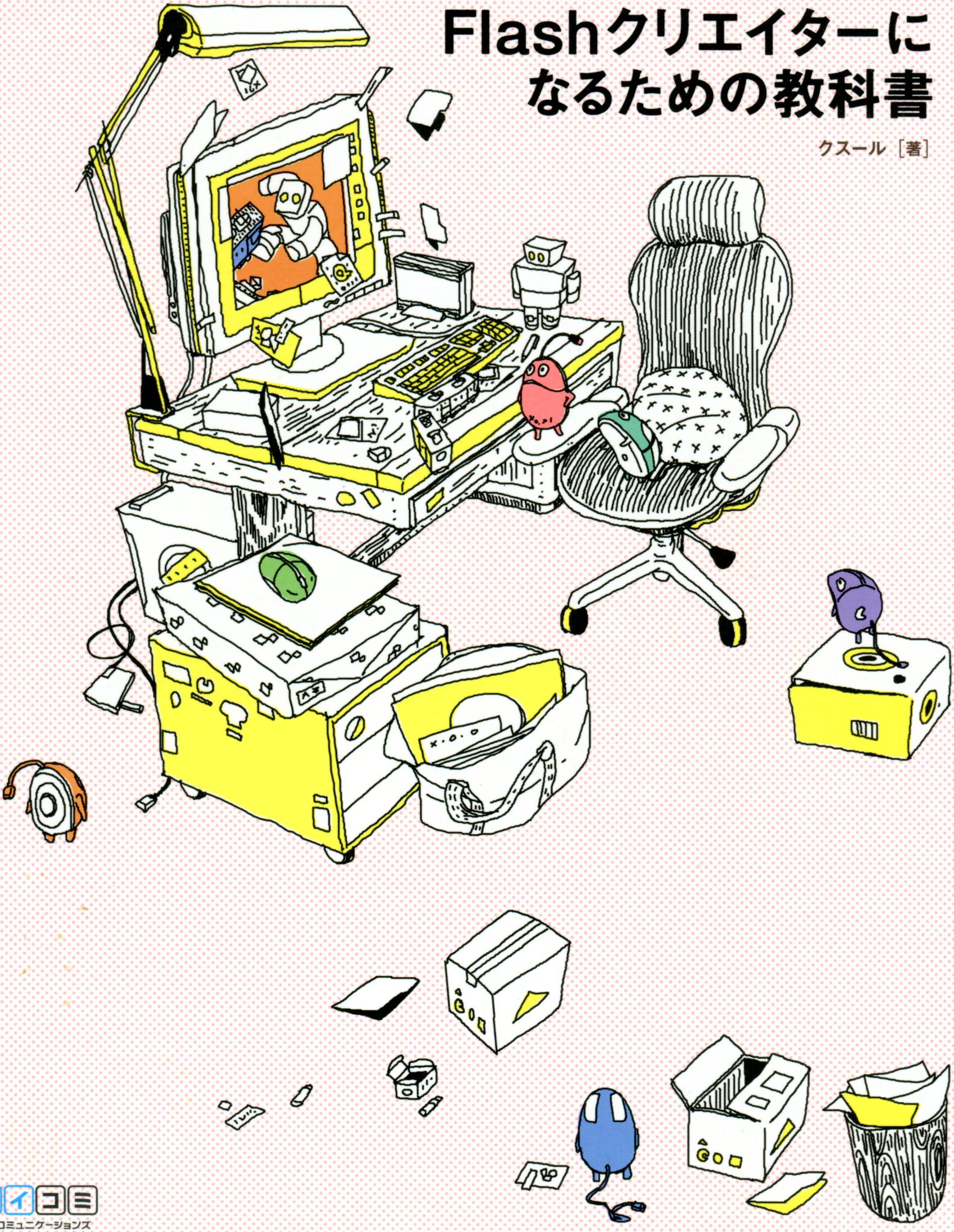
Flashクリエイターになるための教科書

クスール［著］



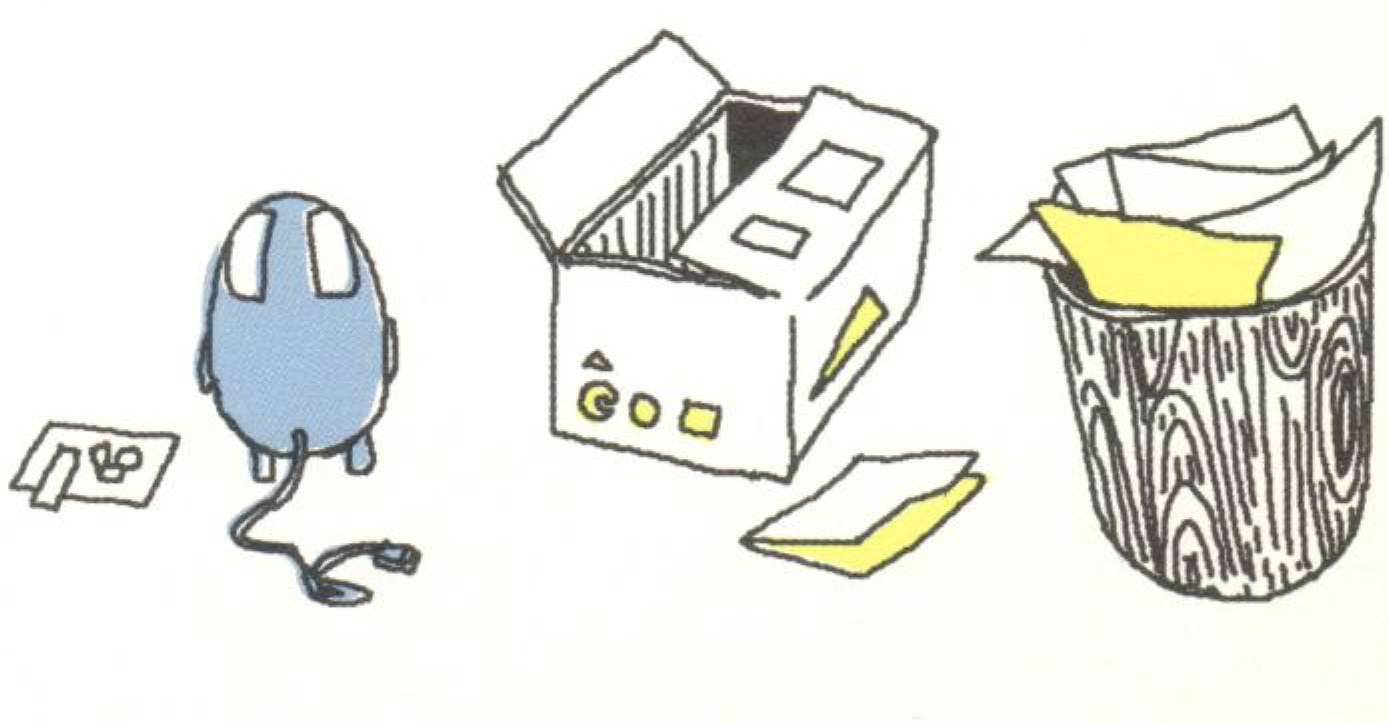
毎日コミュニケーションズ

Flashクリエイターになるための教科書

CONTENTS

Flashクリエイターに
なるための教科書

クスール［著］



マイコミ
毎日コミュニケーションズ

●サンプルファイルの入手先
本書で作成するサンプルのデータを、以下のサイトからダウンロードできます。
またこのサイトでは、本書に関するサポート情報も提供していきますので、ぜひご参照ください。
http://book.mycom.co.jp/support/pc/flash_book/

●本書中の解説はAdobe Flash CS4 Professionalを使用して行っています。

●MacintoshとWindowsとでキー表記やメニューが異なる場合には、双方を併記しています。

●本書は2009年12月段階での情報に基づいて執筆されています。
本書に登場するソフトウェアのバージョン、URL、製品のスペックなどの情報は、すべてその原稿執筆時点でのものです。
執筆以降に変更されている可能性がありますので、ご了承ください。

はじめに

僕たちの会社は平日にウェブ制作の仕事をし、週末にFlashのActionScriptを教えるという、制作と学校の2足のわらじを履いているような会社です。
FlashのActionScriptを教える学校を始めてもう5年以上が経ち、その間にいろんな形式の講座をしました。1日だけのピンポイント技術を教えるもの、1か月間でできるだけ多くの技術を詰め込むもの、2か月かけて受講者に作品を作ってもらうもの、など。人数、時間、机の並びなど、それぞれで全然違う授業になります。
こういった経験の中で、受講者にとって一番良い授業は「受講者自身が手を動かして、作品を作ってもらう」ことだと僕は思っています。
つまりこちらがいくら必殺テクニックやノウハウを教えても、それをコピペするだけでは、その場で達成できただけの技術になります。ところが受講者自身に目標を持ってもらい、それを作ることを課題にすれば、多くの受講者は自分で手を動かし、どんどんとプログラミングしていくようになります。つまり、「技術を学びたい」ではなく、「○○を作りたい」という作品作りのモチベーションが、技術習得には一番近道なのではと感じています。

この本を書くにあたり、授業と同じように読者のみなさんが手を動かして、自分の作品を作っていくような内容にできればと思い、構成を考えました。本書は全部で3つの要素から成り立っています。

PART 1 Flashを知る

ここではFlashで何ができるかを紹介しています。ActionScriptを勉強するにあたり、まずどんなことができるか？　そしてどんなことをしたいかを考えてみてください。そのあとは、PART 2を飛び越し、PART 3で自分の作りたいものに近いものを探してみてください。そこで作りたいものが見つかれば、まずはPART 3をもとにして作品を作っていくことをお勧めします。

PART 2 Flashを学ぶ

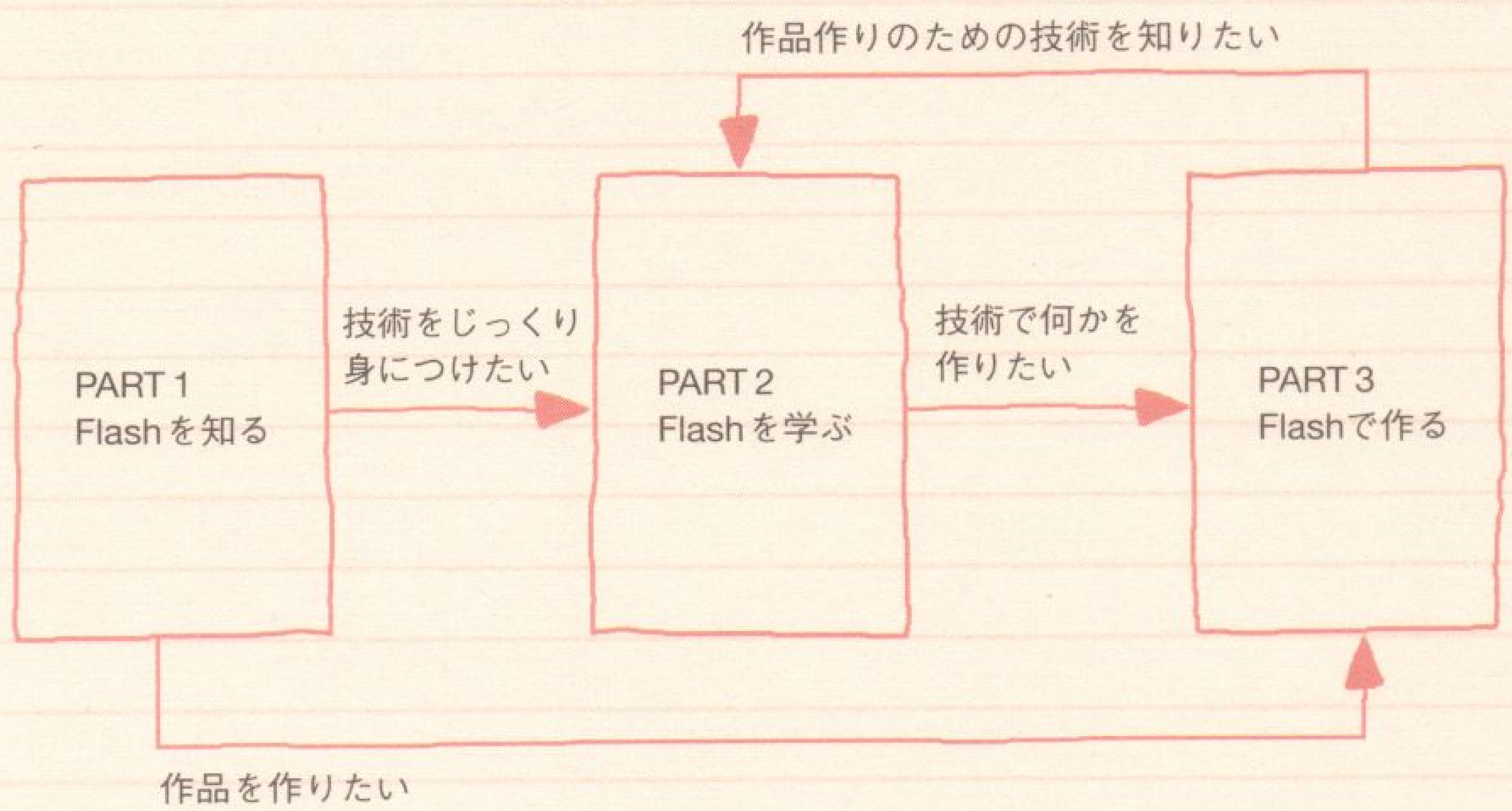
PART 2は辞書・リファレンスのような形式で書いています。ただし、すべてのActionScriptの技術を網羅しているわけではありません。ActionScriptの中には、ある一定以上のものを作るには、何を作る場合でも必ず知っておかなければいけないものがあります。ですので、「必ず知っておかなくてはならない技術（プログラミング基本）」と「作る作品によっては、使わな

くてはならない技術（プログラミング応用）」を分けて書きました。ここは項目ごとがつながっているわけではありませんので、最初のページから順番に学ぶのではなく、PART 3の作品を作っていく際に、分からないことが出てきたらPART 2に戻る方法をとると良いでしょう。

PART 3 Flashで作る

PART 3ではFlashで作品を作るにあたり、多くの人が作ってみたいと思うようなサンプルをピックアップして説明しています。ピンポイントで作りたいものと合致しない場合もあるでしょうが、一通りの技術を網羅していますので、一度自分の作りたい作品に近いものがないかを探し、そこから始めてみてください。また、技術的に複雑なサンプルもありますので、誌面にすべての手順を載せるのではなく、重要なポイントをピックアップして書いています。サンプルファイルをダウンロードして、サンプルを開きながら本書を読んでいくことをお勧めします。

以下に今回の本の読み進め方を図にしてみました。



この本を読んで、もし分からないことがありましたら、一度クスールの教室に遊びに来てください。この本を購入されたのも何かのご縁ですので、気軽にご相談していただければと思います。

2009年12月
株式会社クスール　松村慎

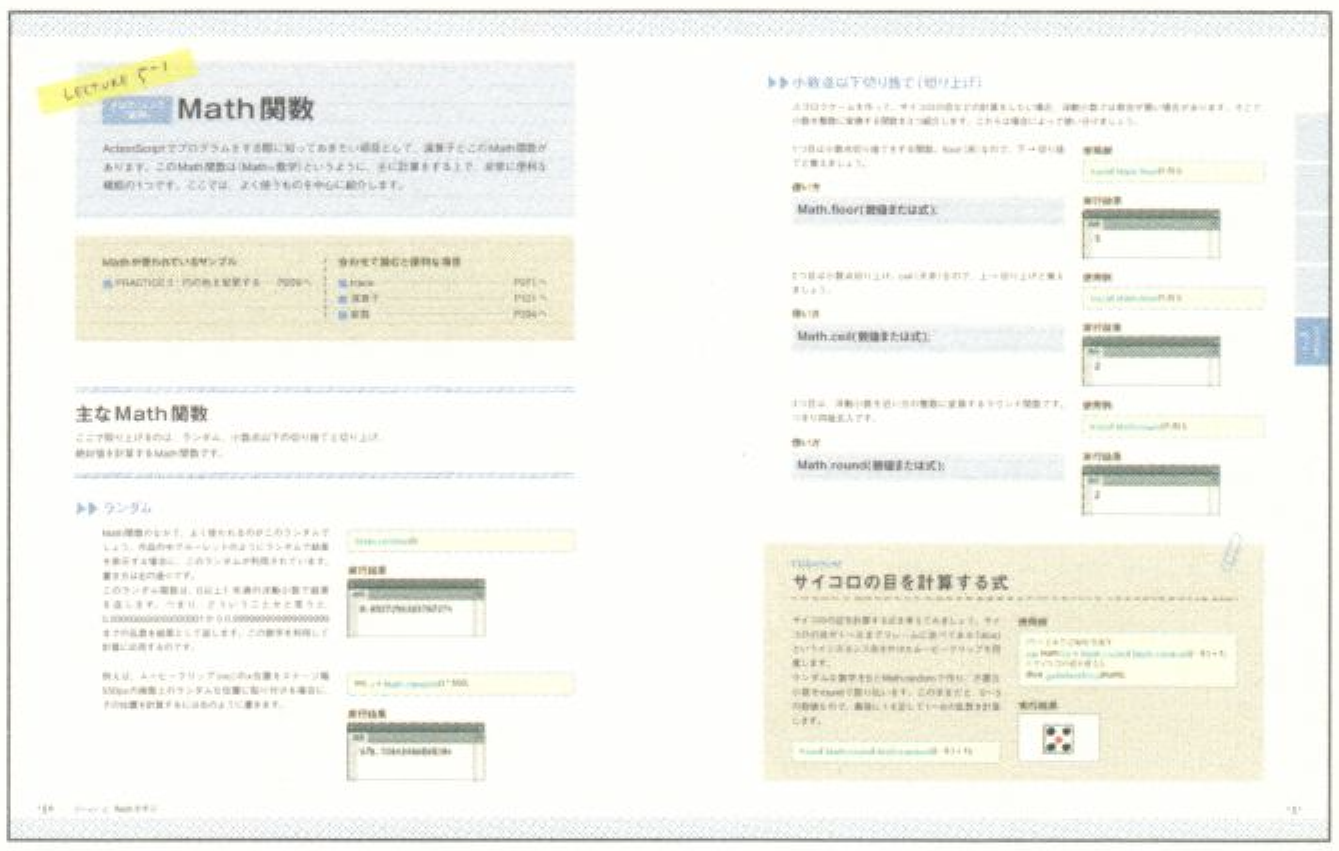
本書の使い方

本書のメインであるPART 2とPART 3について、使い方を説明します。

PART 2　Flashを学ぶ

Flashで作品を作っていくのに必要な知識を簡潔にまとめてあります。

Flashの基本…インターフェイスの説明など、基本的な項目です。
Flashによる表現…モーショントゥイーン、マスク、レイヤーなどについて解説しています。
ActionScriptの考え方…Flashでの作品作りにActionScriptの理解は必須です。ActionScriptの基本を解説しています。
プログラミング基本…変数、イベントリスナ、ifなど、ActionScriptプログラミングで必ず知っておかなくてはならない項目を厳選して解説しています。
プログラミング応用…Math、Timerなど、コンテンツによって必要となってくる項目をまとめています。



それぞれの項目には、その項目を使っているPART 3のサンプルへの参照や、
合わせて読むと理解が深まるPART 2の項目への参照を入れています。勉強の際に参考にしてください。

PART 3　Flashで作る

アニメーションからゲームまで、様々なサンプルを掲載しています。
それぞれのサンプルは次のような流れになっています。

このサンプルで必要な主な基礎技術…PART 2「Flashを学ぶ」の関連項目へリンクしています。サンプル実習の際に、合わせて読むと効果的です。
このサンプルで学ぶ応用技術…サンプルを制作することで何が学べるかをまとめています。制作後の確認にも使えます。
サンプルの説明…ここで学ぶサンプルがどういう動きをするかを解説します。
サンプルの構造…ここで学ぶサンプルのFlashファイルがどういう構造になっているかを図解します。
STEP1～…サンプル制作の手順を解説します。一緒に作ってみましょう。
EX.1～…サンプルに機能や効果を付加する応用テクニックを紹介します。



PART 2とPART 3を交互に参照しながら、効率よく学習を進めてください。

CONTENTS

PART 3 Flashで作る …… 173

PART 1

Flashを知る

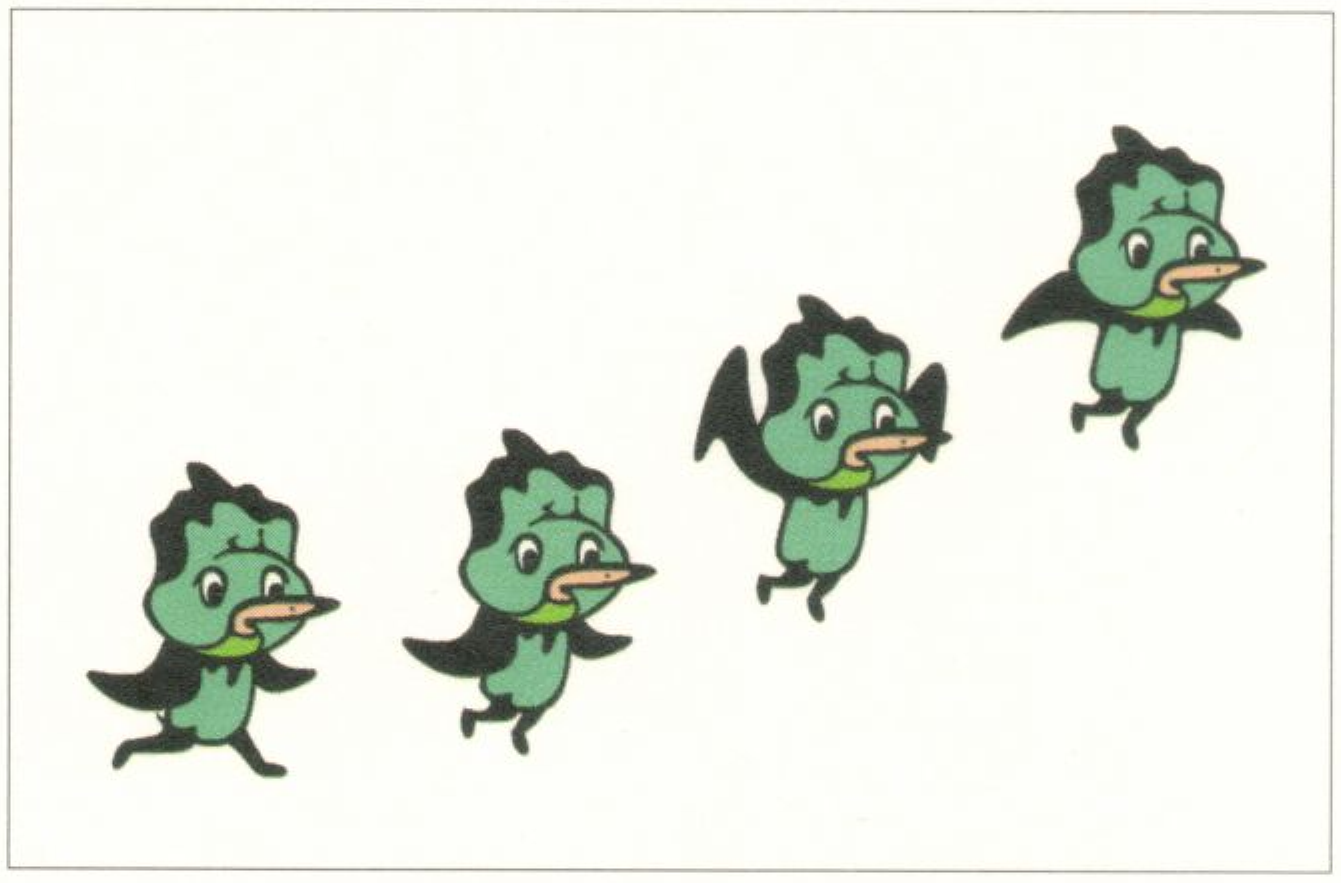
このPARTではFlashの学習を始める前にFlashの歴史、Flashをとりまく技術など、単にツールやプログラミングの技術習得だけでなく、Flashをとりまく環境がどんなものなのかを見ていきます。あくまで著者の経験に基づいて書かれていますので、これがすべてではありませんが、Flashというものを大きな視点から見て、みなさんがどこからどのように始めていくのが良いかのヒントになればと思っています。

THEME 1

Flashでできること

まずはFlashで作ることができるものを少し列挙したいと思います。
技術的に見るとアニメーションとプログラミングという2つの機能しかありませんが、
作る作品で考えると、予想以上に多岐にわたっている印象を受けると思います。

アニメーション



Flashというのはもともとはアニメーションが簡単に作れるソフトでした。一般的に今でもFlashはアニメーションのソフトのイメージが強いでしょう。ActionScriptをあまり使わなくても作れるので、初心者の方には比較的作りやすいです。バナーもFlashのアニメーションをよく使いますが、画面サイズやフレームレートが決まっていたり、短い時間で内容を表す必要があるので、通常のアニメーションを作るのとは別のスキルや経験が必要になります。

■ アニメーションの作り方は……………… P174へ

ウェブサイト

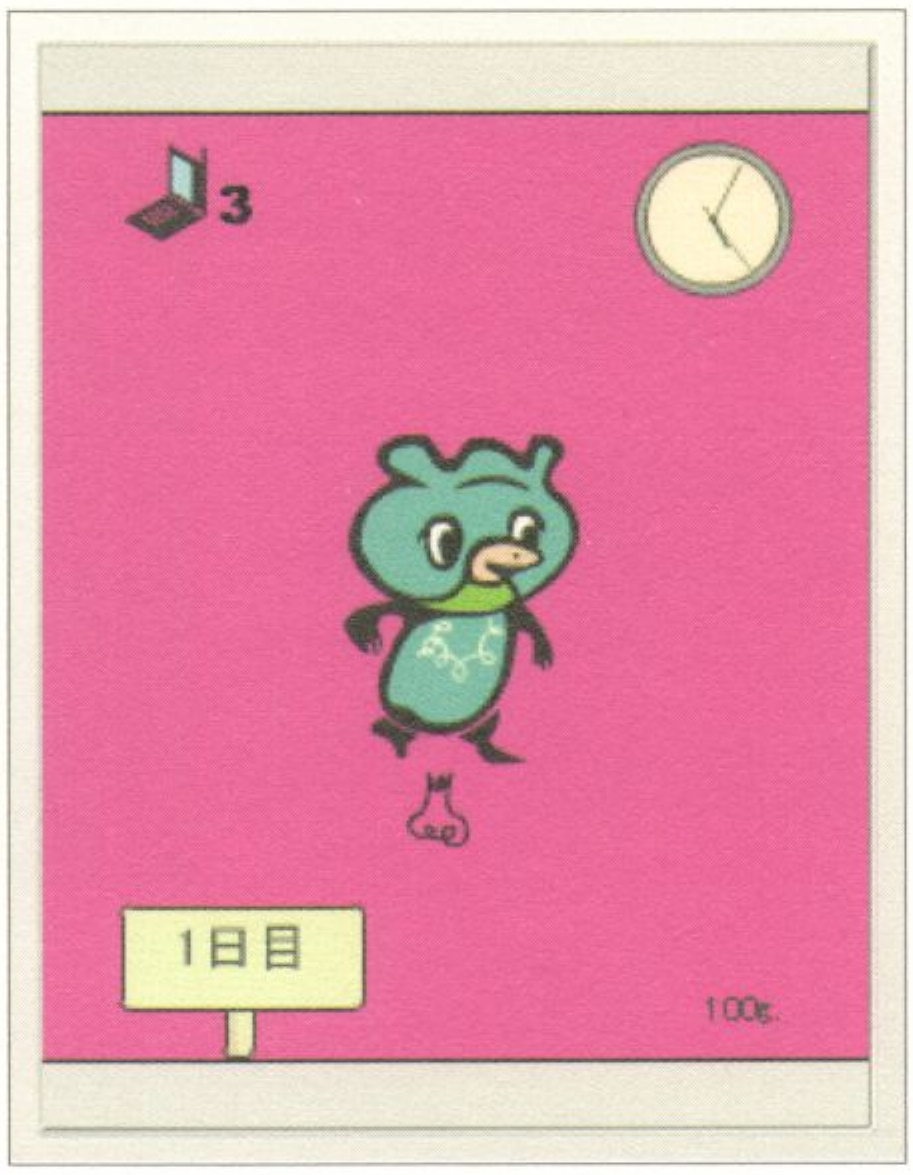


ウェブサイトにはメニュー部分だけFlashで作るものと、画面全体をFlashで作るものに大きく分けられます。
メニュー部分のみの場合はマウスがボタンの上にロールオーバーした場合の動きや、クリックしたときのページの移動先を作るだけなので、アニメーションに簡単なActionScriptを付け加えるだけで作れます。
かたやフルFlashサイトになると、画面全体の設計や、外部ファイルの呼び出しなど、比較的高度なActionScriptのレベルが要求されます。

■ メニューの作り方は………………………… P188へ
■ サイトの作り方は…………………………… P256へ

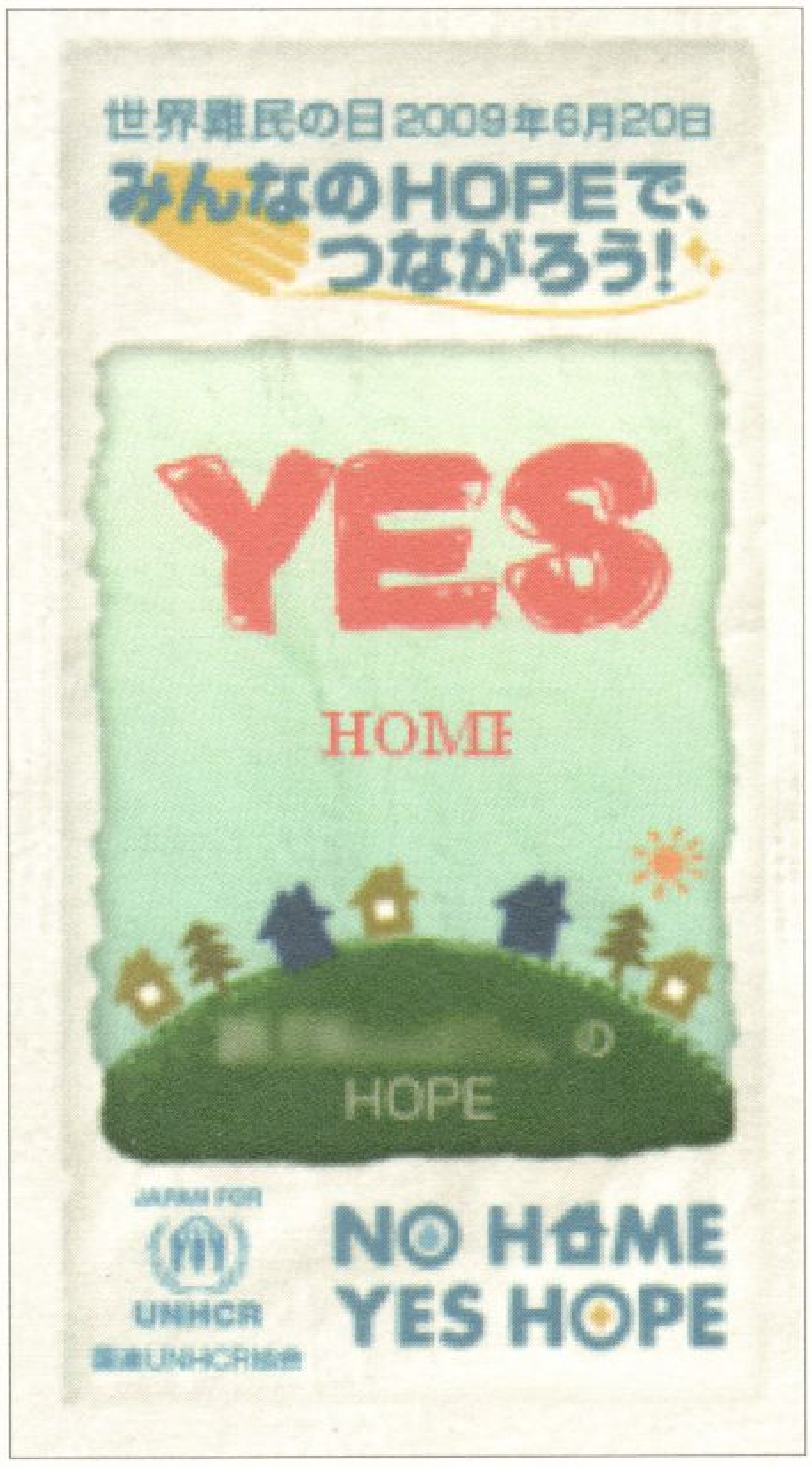
携帯コンテンツ

以前は携帯端末のスペックが低かったり、携帯端末で動くFlash Player（Flash Lite）のバージョンが低かったので、待ち受け画面など簡単なコンテンツを作ることが多かったのですが、最近はFlashメニューやゲームなど、比較的複雑なコンテンツを作ることが多くなりました。
特にFlash CS5からはiPhoneアプリへの書き出しができるようになる予定なので、iPhoneも含め、携帯端末のコンテンツを作る機会は今後増えてくることでしょう。



ブログパーツ

ブログパーツとはブログのメニューやエントリーの部分に表示されるFlashのことです。
HTMLのソースを自分のブログに貼り付けるだけでFlashが表示されるので、その気軽さが受けて、数年前に大きく浸透しました。
今は、はやりと言うよりはすでにスタンダードな存在になっているのではないかと思われます。



ビデオを使ったコンテンツ

Flashを使ったビデオを流すコンテンツとしてはYouTubeが一番有名ですが、Flashでのビデオの特徴は、枠に収まらない使い方ができるということです。
通常のウェブサイトの中の登場人物だけビデオにして、画像と映像とを組み合わせるといったことが可能になります。

■ビデオコンテンツの作り方は………… P256へ



ゲーム

PlayStationやWiiなどのゲームと比べると、Flashでのゲームはグラフィックの表示方法や、計算処理が劣っているため、それほど複雑なゲームは作れません。
ただし、最近はブラウザ上で気軽に遊ぶゲームも一般的になってきているため、Flashゲームも需要が高まっていると思われます。
特にmixiアプリやFacebookアプリなど、いわゆるソーシャルアプリと言われているものの多くがFlashで作られているので、今後はソーシャルアプリの開発需要も増えてくることでしょう。

■簡単なゲームコンテンツの作り方は… P270へ



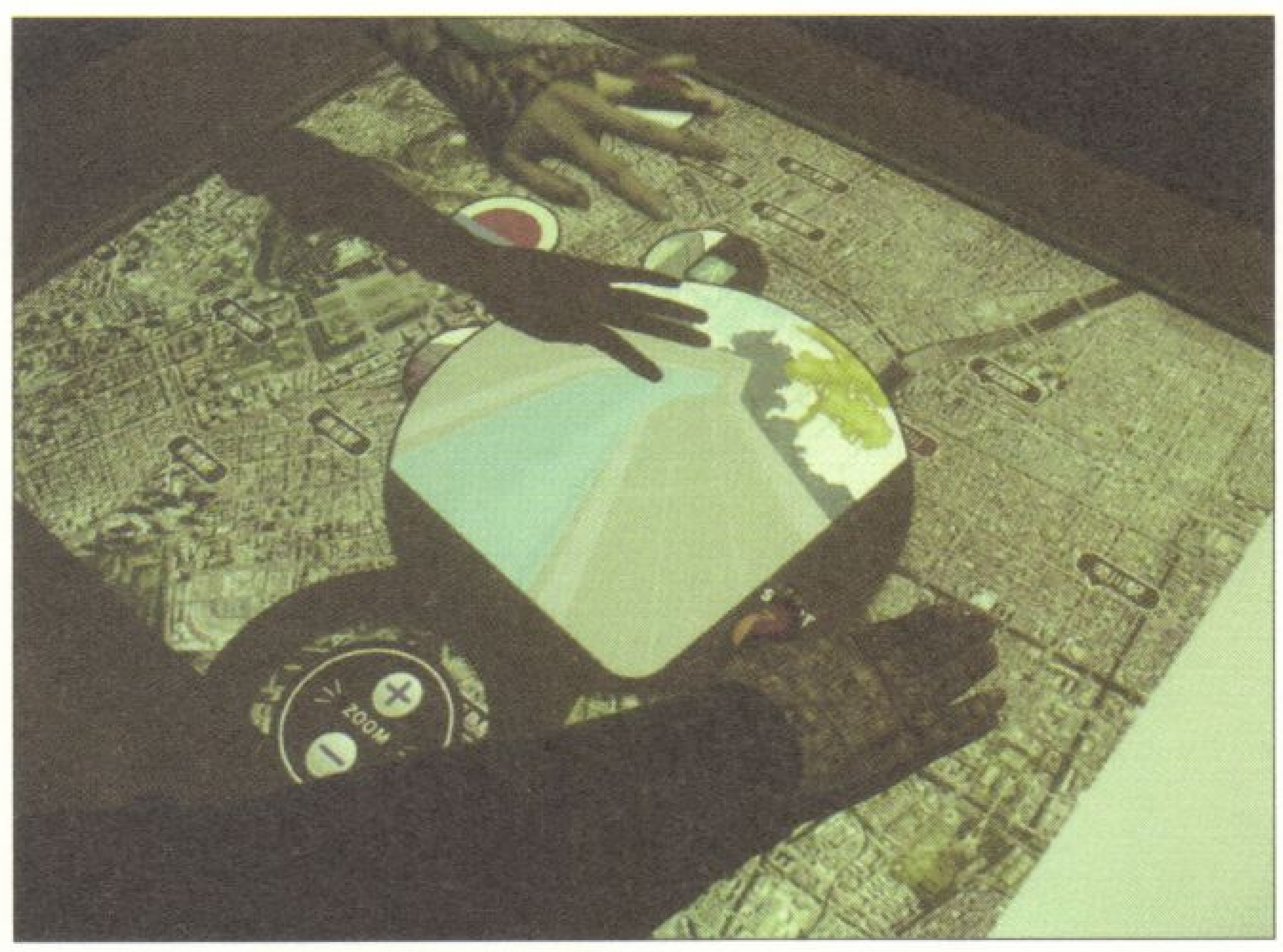
アプリケーション

アプリケーションと聞くとぴんとこないかもしれないですが、普段みなさんがPC上で使っているメールを書くソフトや、ウェブをブラウズするソフトがそれにあたります。
Flashでもこのようなアプリケーションを作ることができます。
Flashでアプリケーションを作るメリットは、「ブラウザ上で使えること(つまりネットにつなげばどのPCでも使えること)」と「通常のアプリケーションよりも違った見た目を表現できること」です。
アプリケーションは今後デザインや使いやすさが重要になってくると予想されるので、Flashの重要性も高まってくることでしょう。



デジタルサイネージ

デジタルサイネージとは、街中でモニタに映像を流したり、タッチパネルで操作したりして、広告や案内をするものです。
今までポスターや案内板の紙媒体を使っていたのが、最近ではデジタルにとって変わってきています。ここでもFlashの需要は増えてきています。2009年に発表されたWindows 7もタッチパネルの機能が備わっていますが、タッチパネルを使ったFlashのコンテンツも今後増えてきそうです。



ということで、ざっとですがFlashでできることを紹介しました。こうやって見ると、Flashでできることが非常に広範囲なのが分かるのではないでしょうか？　つまりFlashでできることはウェブ上のアニメーションやウェブサイトだけでなく、デジタルのコンテンツが動くものすべてにFlashのコンテンツが動作することが可能だということです。まずはFlashの習得を始める前に、どんなものができるか可能性を探ってみるのも良いでしょう。

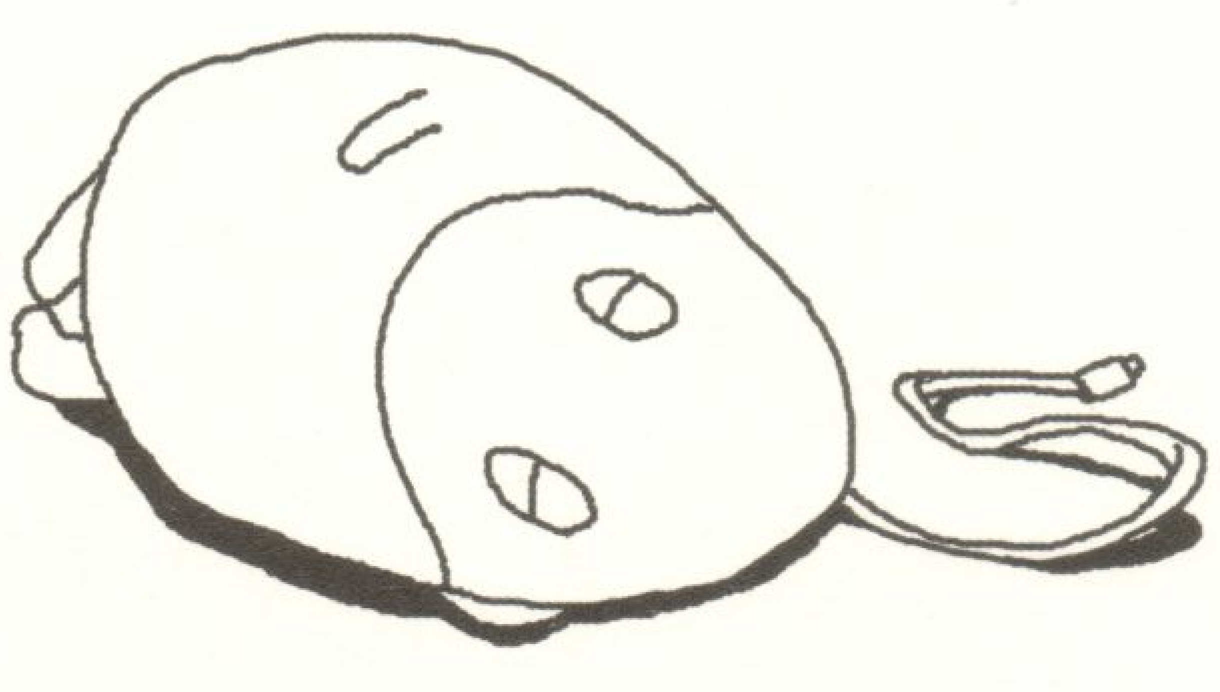
THEME 2

Flashの歴史

Flashはもともとアニメーションツールですが、THEME 1で紹介しているようなプログラミングや機能がどのように追加されていったのでしょうか？
下の表を見てください。

年代	Flashのバージョン	FlashPlayerのバージョン	主な追加機能	ActionScriptのバージョン
1996	Flash 1	Flash Player		
1997	Flash 2	Flash Player 2		「アクション」搭載
1998	Flash 3	Flash Player 3		
1999	Flash 4	Flash Player 4		
2000	Flash 5	Flash Player 5	XMLの読み込み	「ActionScript」搭載
2002	Flash MX	Flash Player 6	ビデオの再生 Flash Communication Serverの発表	
2003	Flash MX 2004	Flash Player 7		「ActionScript 2.0」搭載
2005	Flash 8	Flash Player 8	ドロップシャドウなどのエフェクト	
2007	Flash CS3	Flash Player 9		「ActionScript 3.0」搭載
2008	Flash CS4	Flash Player 10	新しいモーショントゥイーン ボーンアニメーション	

これを見ると分かるのですが、プログラミングのバージョンアップとツールの機能が交互に進化していっています。
これは意図してこのように追加しているのかは分かりませんが、このようにFlashでできることが増え、さらにそれを支えるプログラミングがバージョンアップすることで、Flashがアニメーションツールからさまざまなものを扱えるツールに進化していったのが分かることでしょう。

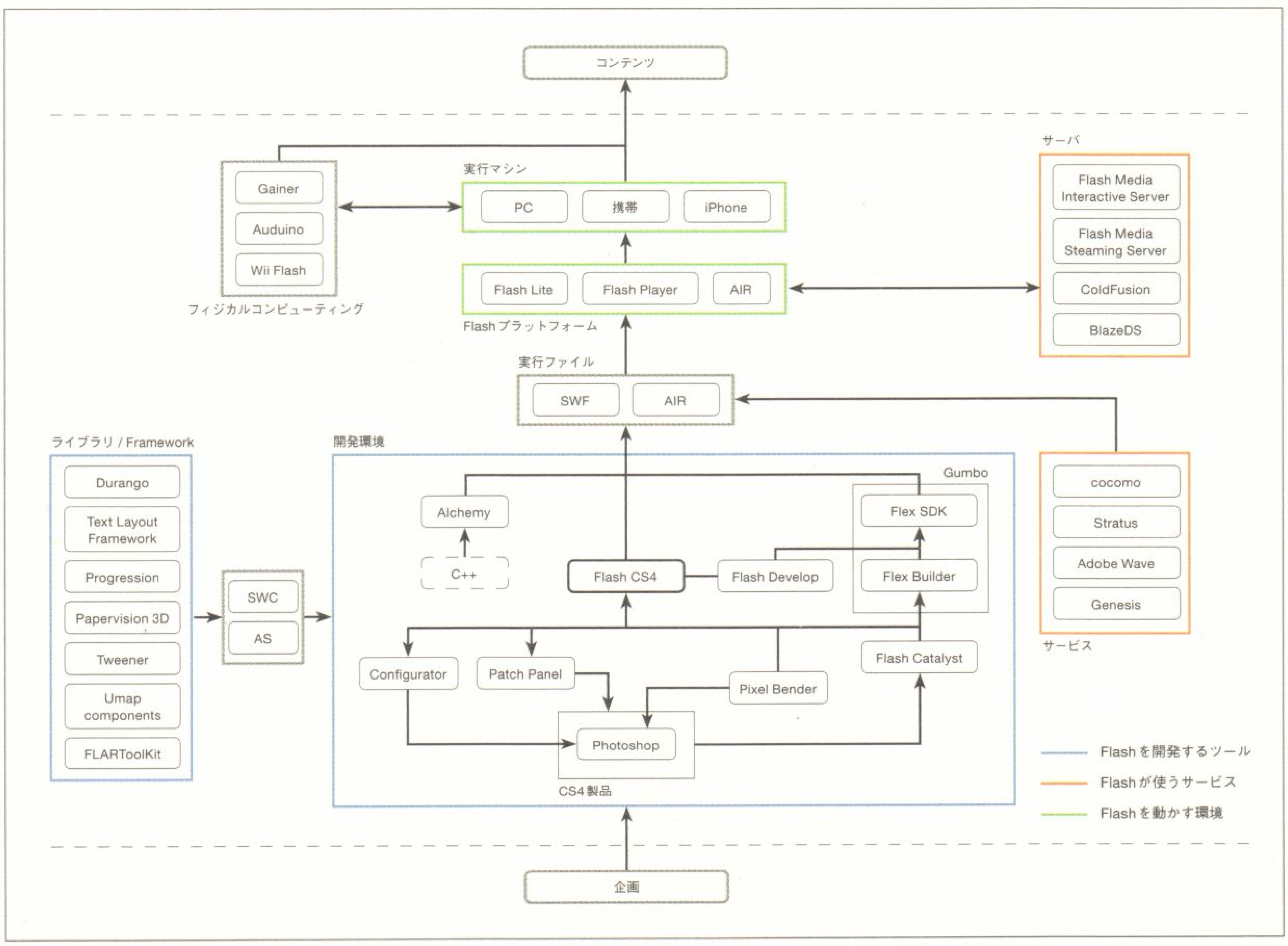


THEME 3

Flashを取り巻く技術

Flashを再生するコンテンツはSWFファイルと言いますが、このSWFファイルが作成できるのは、Flashのオーサリングツールだけではありません。Flash BuilderやFlex SDKなど別のツールや技術を使っても作成できます。
またTHEME 1で紹介したゲームやアプリケーションを作成するにはFlash以外にサーバに接続したり、他の言語と連携することもあります。そういう意味でFlashで複雑なコンテンツを作るにはFlash以外の技術も必要になります。
今の時点で発表されているFlashをとりまく技術を少し表にまとめてみました。
これらはAdobeがリリース前の技術やアプリケーションを紹介するAdobe Labs（http://labs.adobe.com/）というサイトで公開されている情報をもとにまとめています。これをすべて紹介するのは無理ですが、下記のようにいくつかのポイントが挙げられます。

- Flashオーサリングツールだけで開発するよりも、Flash DevelopやFlash Builderなど、プログラミング開発ツールと同時に使用することが今後増えてくると思われる。
- ActionScriptでは他の人が開発したライブラリというものが多く出回っているので、すべてを1から開発するより、他の人のライブラリを使って効率良く開発することが増えてくると思われる。
- Adobeがウェブを通じて提供するサービスを増やしてきているので、それらを使う機会が増えてくる。
- iPhoneや携帯など、PC以外で動くコンテンツを作ることが増えてくるので、PCと各デバイスの両方を同時に開発するということが出てくると思われる。



Flashを学ぶ手順

THEME 4

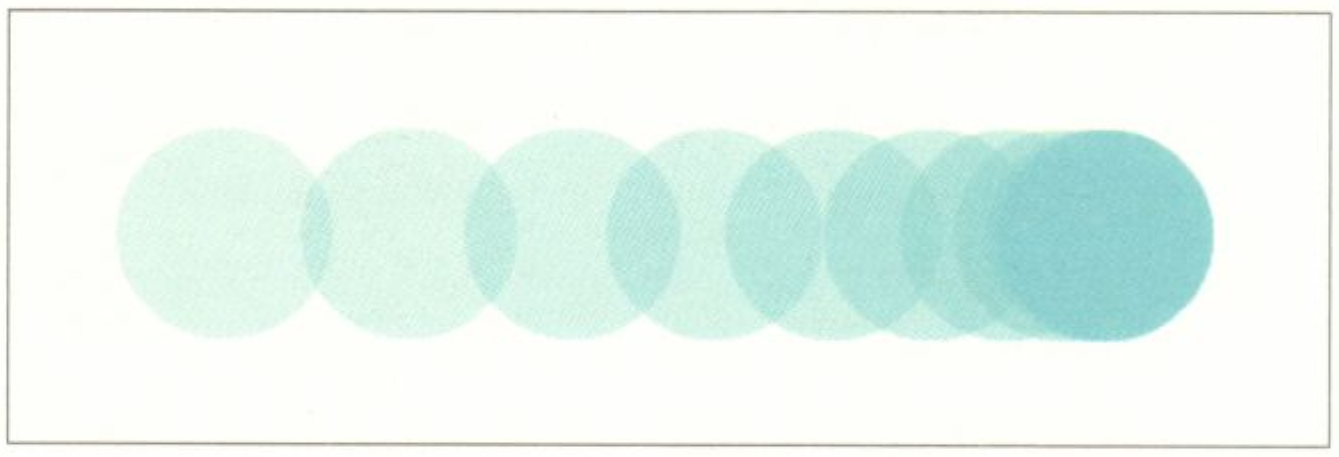
THEME 2でFlashの歴史を、THEME 3でFlashを取り巻く技術を紹介してきましたが、
実際これらを踏まえてみなさんがどのように技術を身につけるのか？
オーソドックスではありますが、その手順を紹介していきます。

技術として学ぶ手順

まずは技術としてどのような手順で学んでいくのかを紹介します。
この順番はFlashをまったく触ったことがない、プログラミングの知識が無いということを前提にしています。

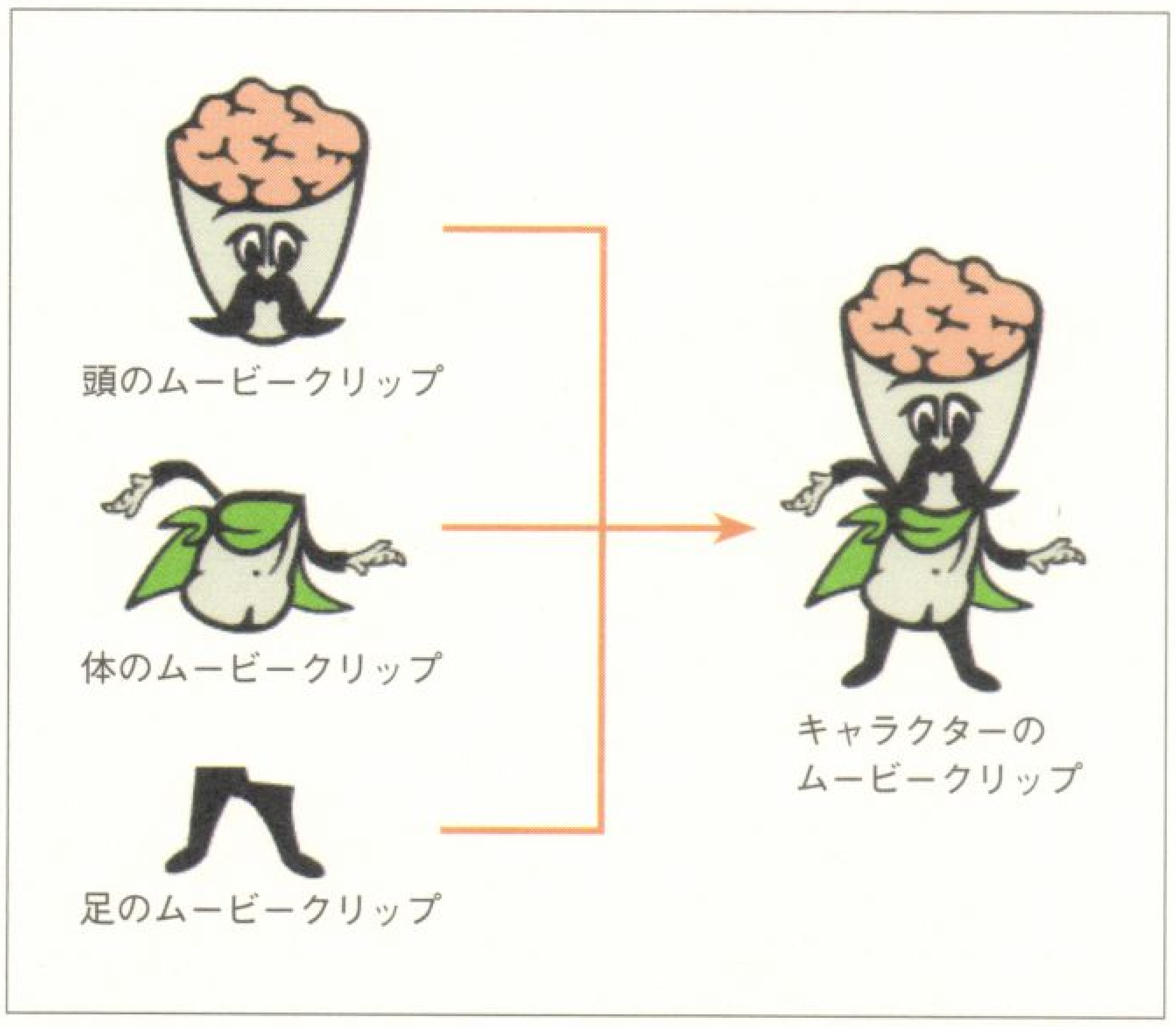
1. トゥイーンを使い、アニメーションの方法を学ぶ

まずは最初にアニメーションを作ってみることをお勧めします。Flashの特徴は直感的に作ったものがそのまま動くことです。まずはこのアニメーションが動く楽しさを実感するのが良いでしょう。



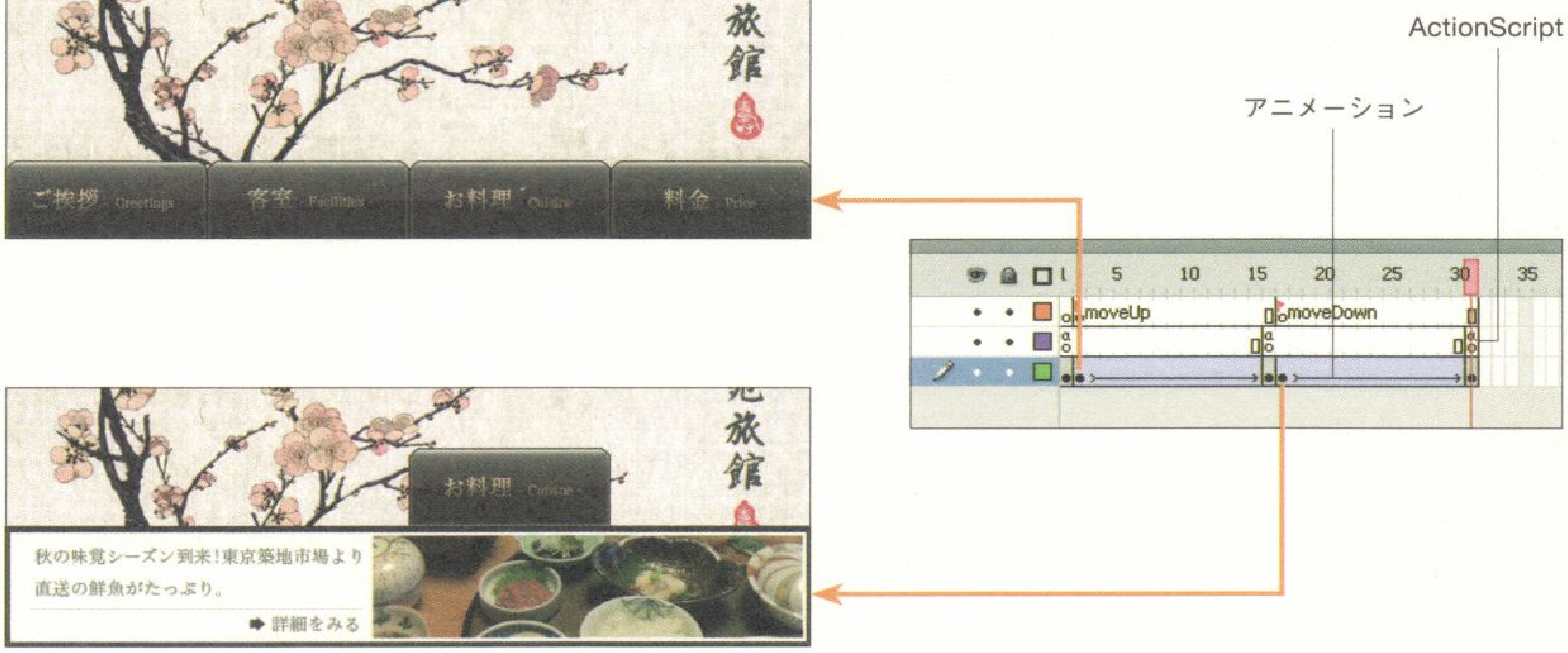
2. シンボルの入れ子構造を使いこなす

Flashでまず最初につまづく部分がムービークリップ内にさらにムービークリップが入っているような入れ子構造を理解することです。
設計次第では入れ子構造を使わなくても良いですが、複雑なアプリケーションになるにつれ、この概念は重要になってくるので、初めの段階でこの仕組みを覚えることが後のスムーズな理解につながります。



3. 簡単なActionScriptとアニメーションを組み合わせる

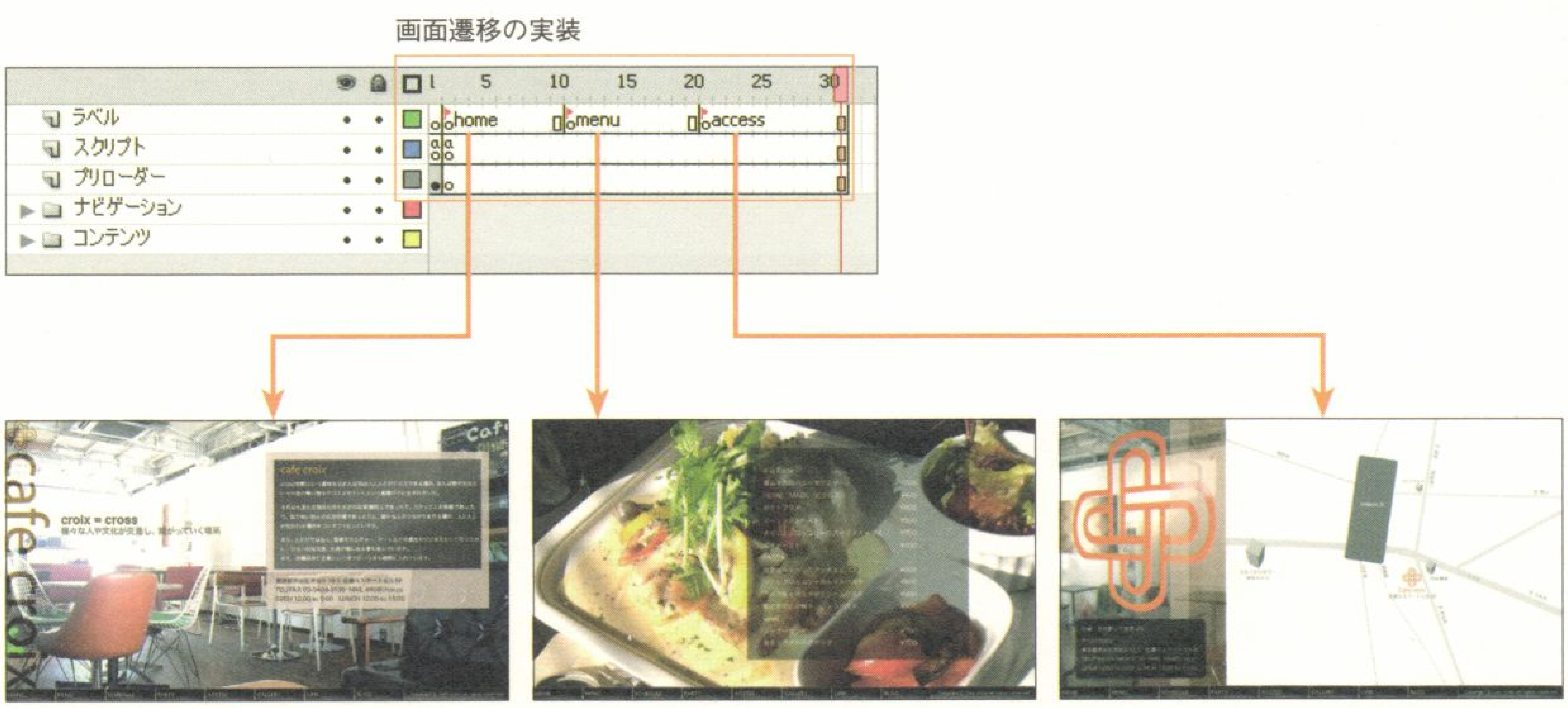
gotoAndPlay、gotoAndStopなどタイムラインをコントロールするActionScriptを使い、アニメーションをコントロールします。これらActionScriptはそれほど使い方が難しくなく、かつ見た目の制御なので、プログラミングが苦手な人もそれほど抵抗なく進めることができるでしょう。



4. フルFlashサイトを作る

フルFlashサイトを作るには少し構造の設計が必要になります。サイトには複数のページがあると思いますが、そのページ移動の際にアニメーションが加わります。

つまり1ページ目→アニメーション→2ページ目という動きが加わるのですが、逆の2ページ目→アニメーション→1ページ目という遷移も必要になります。これらをどのように組み立てるかを考えるだけでも複雑さが分かるでしょう。



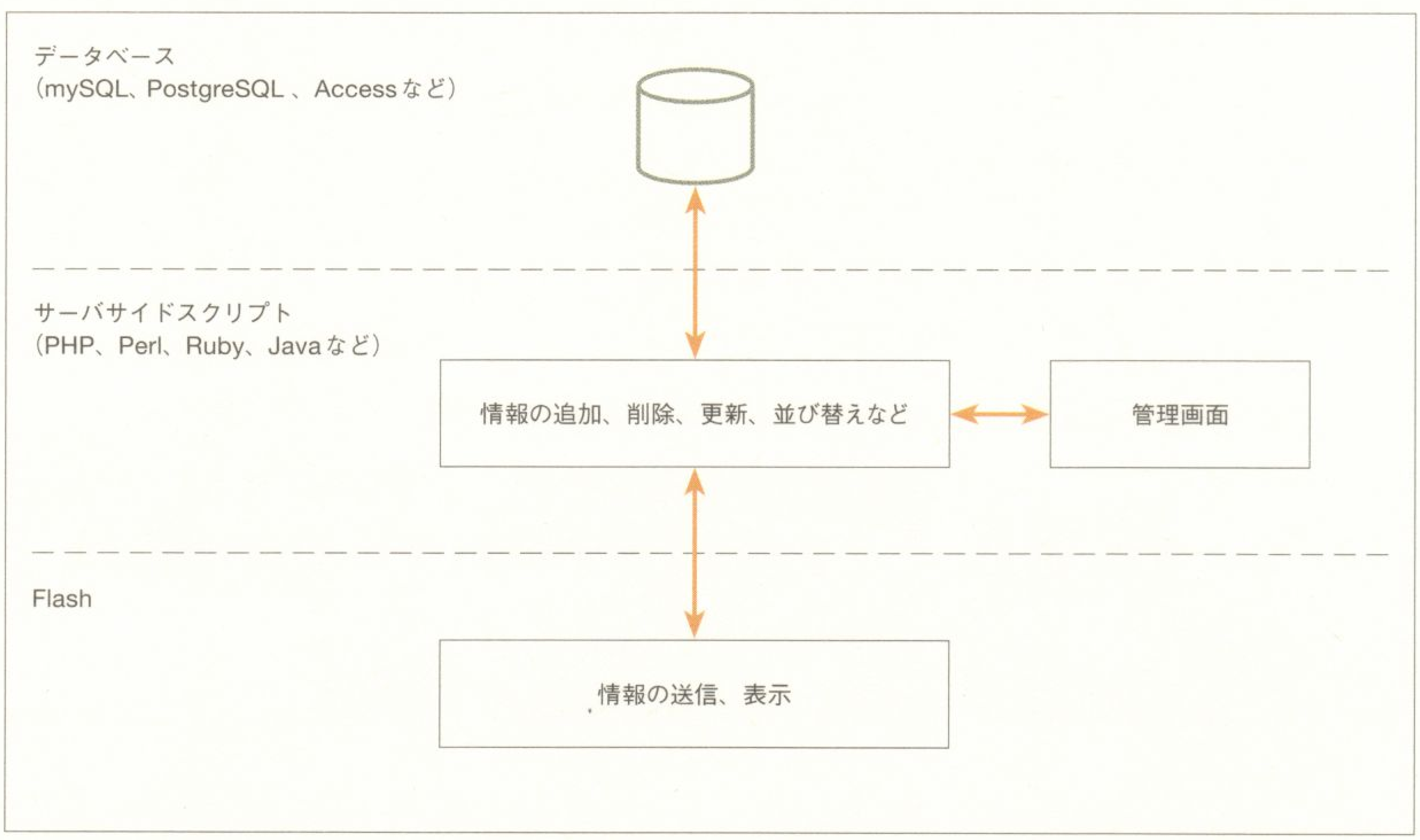
5. 更新性のあるサイトを作る

FlashもHTMLサイトと同じように、コンテンツの更新が頻繁に行われることがあります。
ただ、その際にFlashの内容を毎回書き換えて公開するのはかなりの労力がかかります。これを防ぐために、頻繁に更新する文章や、写真はFlashの外からその都度呼び出すようにします。
この外部からファイルを呼び出すための仕組みをあらかじめ設計する必要があります。



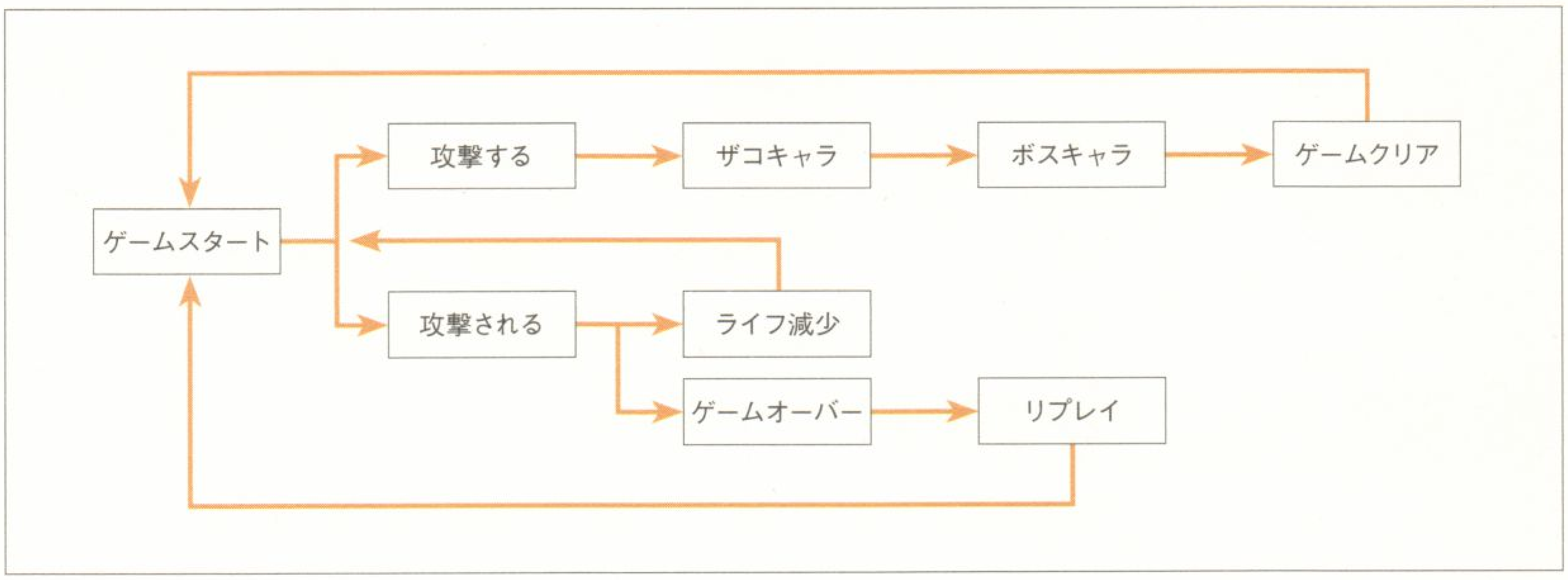
6. サーバの情報と連携する

上記の5.で設定した更新される情報は外部ファイルを書き換えることで更新できます。ただし、その情報をサーバに再度アップする必要があるので、FTPのつなぎ方などウェブの技術を知っている人でないと行えません。
これを解消するために、データベースとサーバサイドのプログラミングが必要になります。
これら技術を使うとウェブブラウザからも情報更新ができ、ウェブの技術をそれほど知らない人でも更新が行えます。



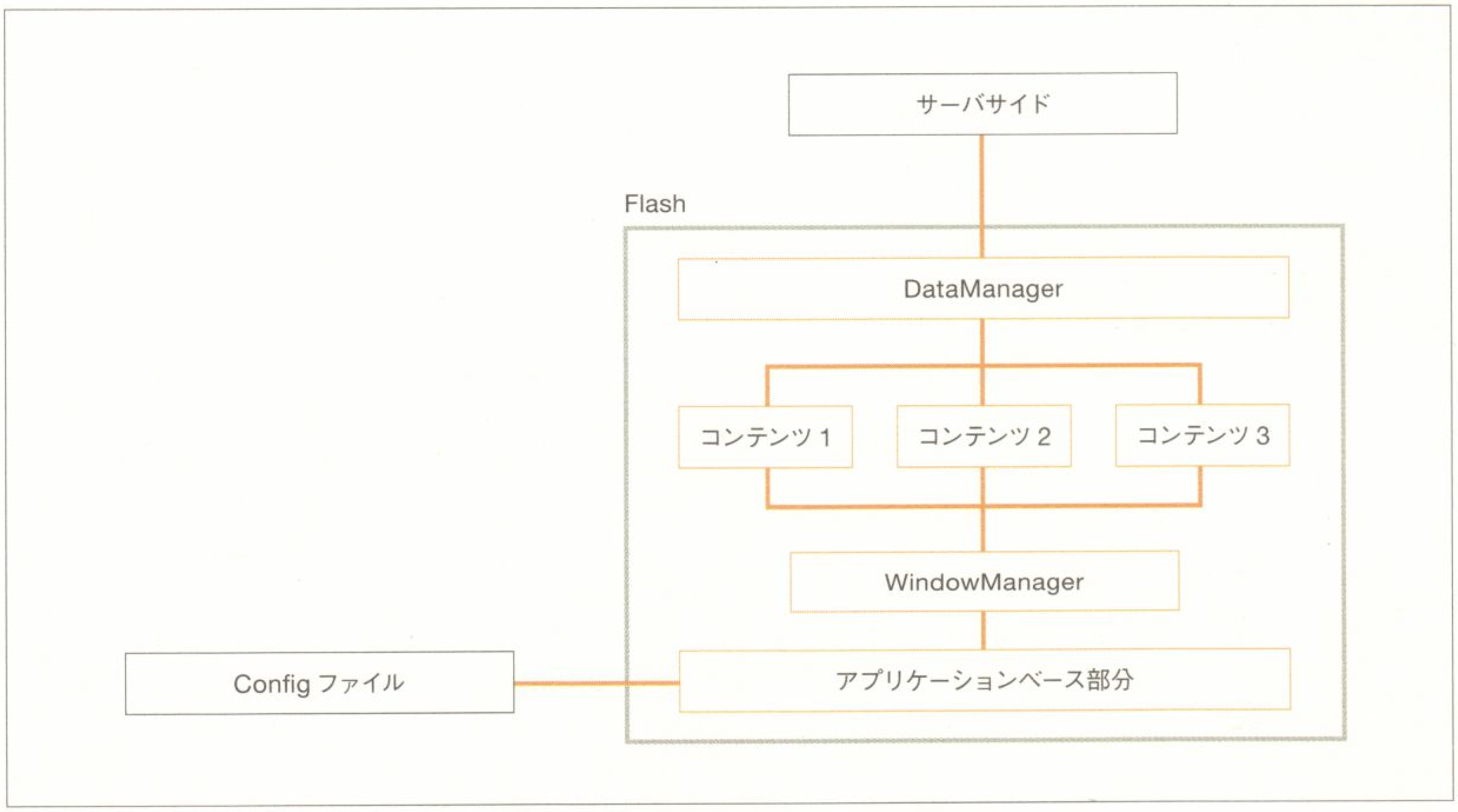
7. ゲームなど複雑なロジックを実装する

サイトとゲームを作るときの大きな違いは、インタラクティブな要素の多さにあると思われます。サイトでは「ボタンが押されたとき」のインタラクションをメインに考えていけば良いのですが、ゲームでは「プレイヤーが攻撃したとき」「敵の弾があたったとき」「プレイヤーのレベルが上がったとき」「ゲームオーバーになったとき」など多くのインタラクションが起こる場面があります。それらをすべて考慮した構造を考える必要があるので、サイトとは違った設計の知識が必要になります。サイトはもうたくさん作ったことがあると言う人は、まずは簡単なゲームを作ることが、実力アップにつながるでしょう。



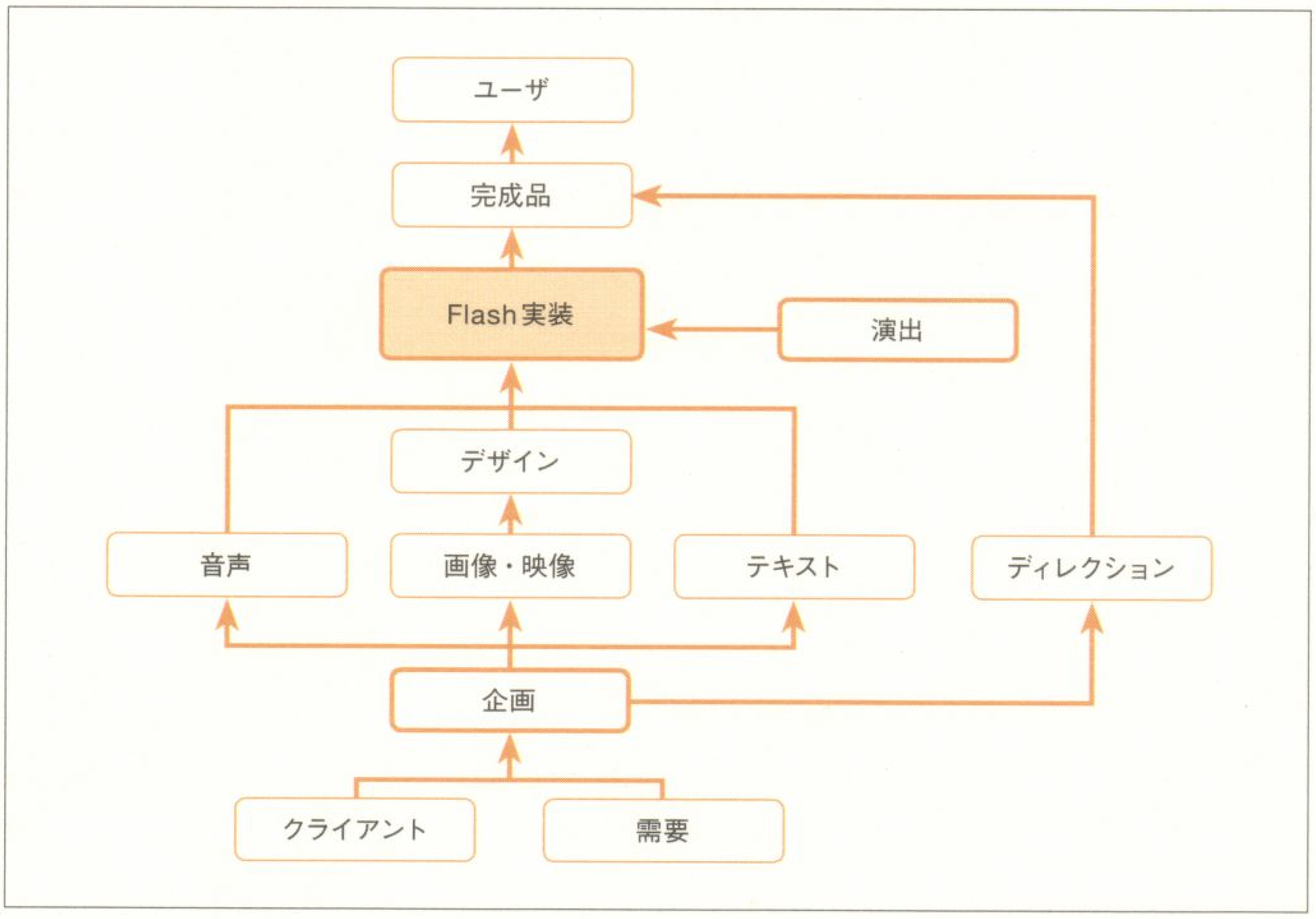
8. アプリケーションの設計をする

アプリケーション作成とゲーム作成はどちらが難しいかは一概に判断できませんが、アプリケーションはデータベースを絡めて作成されることが多く、それらデータの更新なども考慮しなければいけません。
そういった意味で、データベースと絡めたアプリケーションを作れるようになれば、かなりの実力が付いた証でしょう。



企画を絡めたFlashの使い方を考える

今までは技術レベルの話をしてきましたが、Flashのコンテンツ作成には技術以外に、表現やアイデアが大きな要素を占めます。プログラミングとアニメーションをどのように組み合わせるときれいにみえるか。世の中にない面白いコンテンツを生み出すにはどのようなアイデアが必要か。など、技術とは違った方面での工夫がコンテンツをさらに面白くさせることができます。この部分は技術を身につける過程の段階でもぜひ取り組むと良いでしょう。



COLUMN

作品を作ることを念頭におく

THEME 4では技術を身につける順番を紹介しましたが、実際にはまず作りたいコンテンツを想定し、そのコンテンツに必要な技術を身につけていくという考え方がお勧めです。
何を作りたいかの目的なしで技術だけを身につけようとすると、最終的にその技術で何をしたいかが分からなくなることがよくあります。まずはTHEME 1「Flashでできること」で紹介した中で、どんなものを作ってみたいか考えてみてください。その後、それに必要な技術を洗い出し、1つずつステップアップしながら技術を身につけていくのが良いでしょう。
この本のPART 2ではピンポイントに技術を身につける方法、PART 3ではそれらを利用した作品作りの紹介をしています。
まずはPART 3で作りたいコンテンツを探し、そのコンテンツを作るための技術を身につけたい場合はPART 2で調べるといった、PART 2とPART 3を行き来しながらコンテンツを作り上げていくと実力が身に着くと思います。

PART 2

Flashを学ぶ

このPARTではFlashのツールやActionScriptについての使い方を項目ごとに分けて説明していきます。この本の中では辞書的な役割をしています。通常の辞書のようにすべてを網羅しているわけではありませんが、Flashをする際に必ず必要な部分をピックアップし、そこを深く掘り下げて書いています。PART3でサンプルを作る際に、分からない項目が出てきた際は、ここに立ち戻って調べてみてください。

LECTURE 1-1

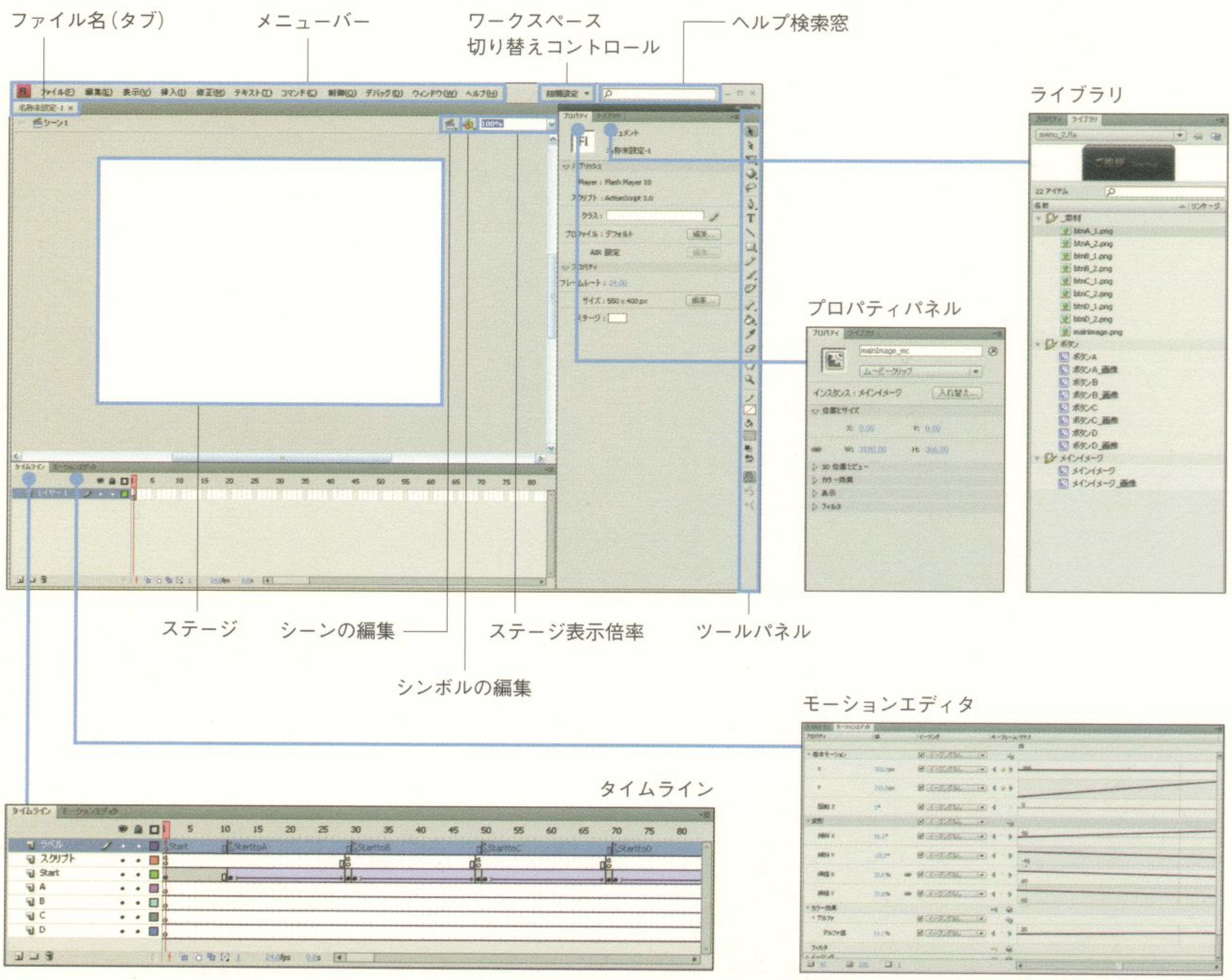
Flashの
基本

Flashの画面構成

Flashの画面構成（ワークスペース）はカスタマイズし、保存することが可能ですが、初期設定の状態では下図のように表示されます。パネルはここに出ている以外でも出力パネル、アクションパネル、カラーパネル、プロジェクトパネルなど多くの種類があるので、色々試しながら自分好みのワークスペースを作ってください。

Flashのインターフェイス

ここでは「初期設定」ワークスペースの画面で解説を行います。
ワークスペースは、[ウィンドウ→ワークスペース] のサブメニューから変更できます。



LECTURE 1-1

ファイル名（タブ）

現在開いているドキュメントのファイル名を表示します。複数のドキュメントを開いているときは、このタブをクリックすることで、編集するドキュメントを切り替えます。

メニューバー

Flashで実行したい機能を選択するためのメニューが格納されています。

ワークスペース切り替えコントロール

登録／保存してあるワークスペースの切り替えができます。アニメーター、クラシック、デザイナー、デベロッパーなど、いくつかのデフォルトのワークスペースがあるほか、カスタマイズしたワークスペースを保存しておき、呼び出すこともできます。

ヘルプ検索窓

ここに入力したキーワードで、Flashのヘルプを検索することができます。CS4から［ヘルプ］パネルがなくなり、オンラインヘルプを参照する形になりました（オフライン環境下では、ローカルに保存されているHTMLドキュメントを表示します）。

シーンの編集

シーンを分けてドキュメントを編集している場合は、ここで編集するシーンの切り替えができます。

シンボルの編集

ボタンをクリックすると、ドロップダウンでライブラリのシンボルが表示され、それぞれのシンボル編集画面にジャンプできます。

ステージ表示倍率

ステージの表示倍率を変更できます。ドロップダウンメニューから選択できる数値のほか、直接数値を入力することもできます。

ツールパネル

Flashで使用する様々なツールがここに格納されています。CS4ではツールパネルは初期設定では右側に配置されています。

■ツールの使い方 ………………………… P029へ

プロパティパネル

現在選択されているオブジェクトやドキュメントのプロパティが表示されます。プロパティの変更もここで行うことができます。

ライブラリ

ドキュメントで使用できるアイテムが表示されています。Flash上で作成したものをアイテムとして保存できるほか、他のアプリケーションで作成した画像、ベクターデータ、映像、音声などをアイテムを読み込むことも可能です。

タイムライン

時間の長さをフレームに分割し、コンテンツをフレーム単位で表示しています。

モーションエディタ

Flash CS4から搭載された機能です。モーショントゥイーンを作成した際に、アニメーションさせているシンボルのx, yや回転、傾斜、アルファなどを細かく調整することができます。

LECTURE 1-2

Flashの基本

ファイルの作成とパブリッシュ

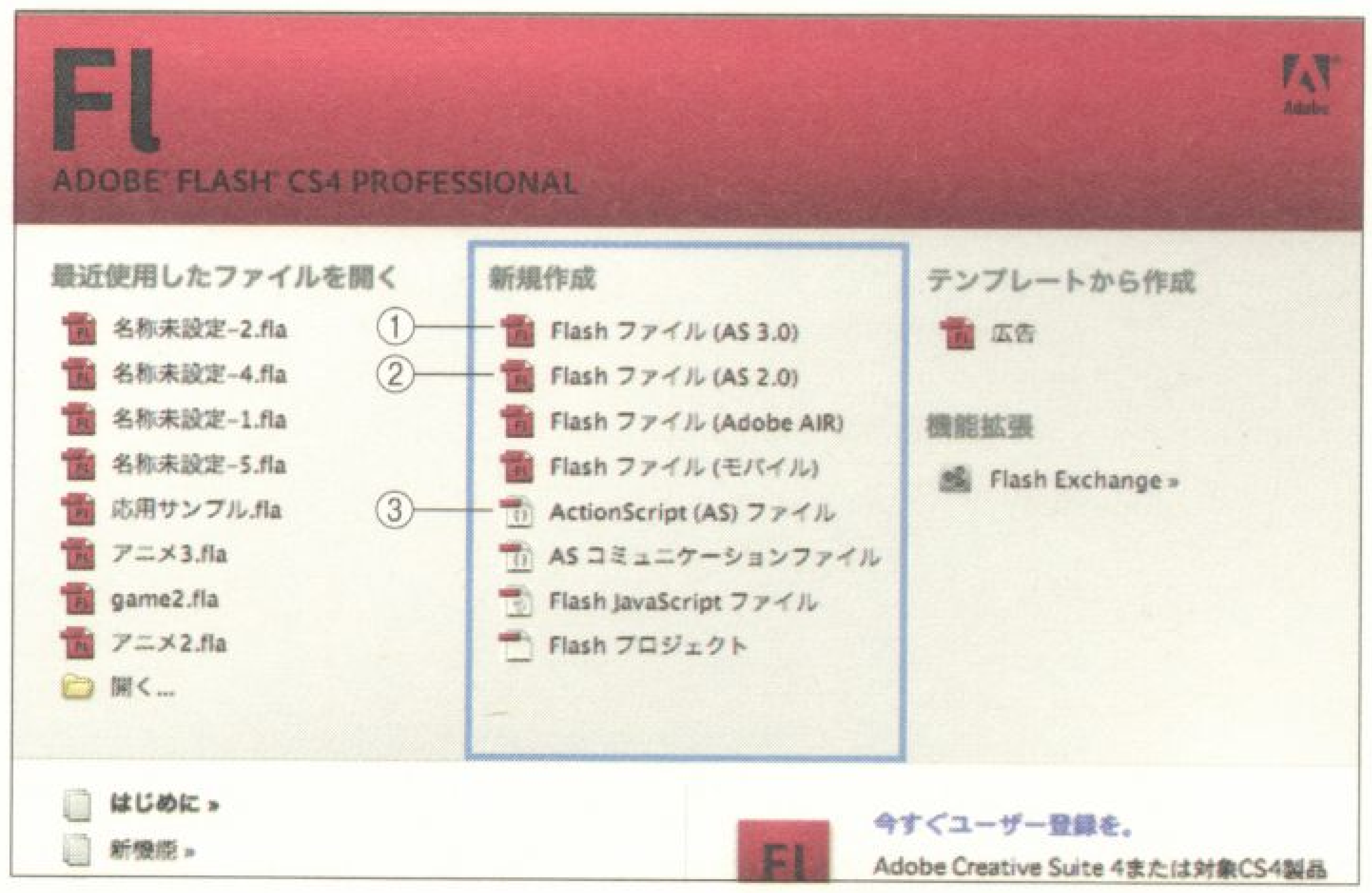
Flashファイルを作成する際にまず、ファイルの大きさや、アニメーションの速度を決めるフレームレートを設定する必要があります。ファイルの設定はプロパティパネルで行うことができ、常に変更することが可能ですが、始めに設定しておくことで、効率良く作業を行うことができます。

新規ファイルの作成

Flashを新規で作成していきましょう。
Flashにはさまざまな形式のファイルがあるので、始めに使用したい形式を選択する必要があります。

▶▶ 新規ファイルの作成

Flashを立ち上げるとデフォルトでスタートページが表示されます。
「新規作成」で使用する形式を選択します。



①Flashファイル(AS 3.0)

スタートページの「新規作成」列の「Flashファイル(AS 3.0)」をクリックし、空白のドキュメントを作成します。
AS 3.0は、Flash Player 9以降のプラグインで実行できるプログラムを作ることができる、オブジェクト指向スクリプト言語のことです。

②Flashファイル(AS 2.0)

スタートページの「新規作成」列の「Flashファイル(AS 2.0)」をクリックし、空白のドキュメントを作成します。
AS 2.0は、Flash Player 7以降のプラグインで実行できるプログラムを作ることができる、オブジェクト指向スクリプト言語のことです。

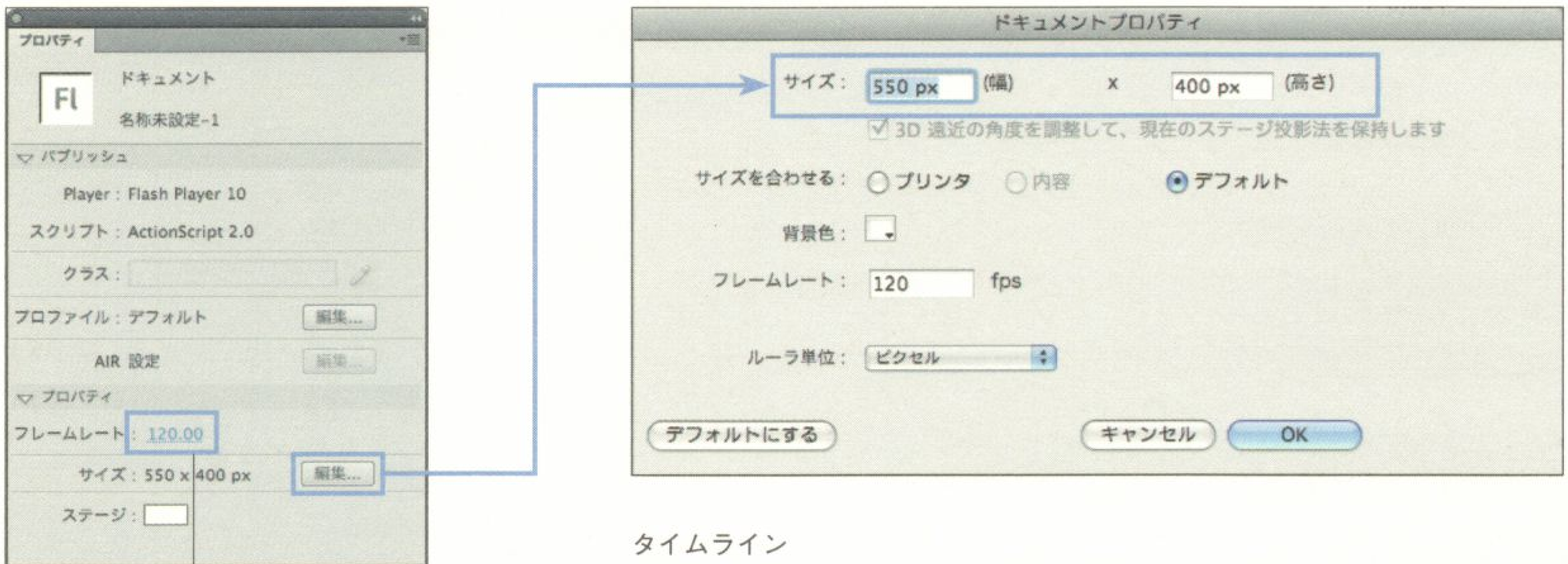
③ActionScript (AS) ファイル

FlashのActionScriptを外部ファイル(asファイル)に記述する際に使用します。複数人で分割作業をするときなどにも便利なファイルです。

ステージサイズとフレームレートの設定

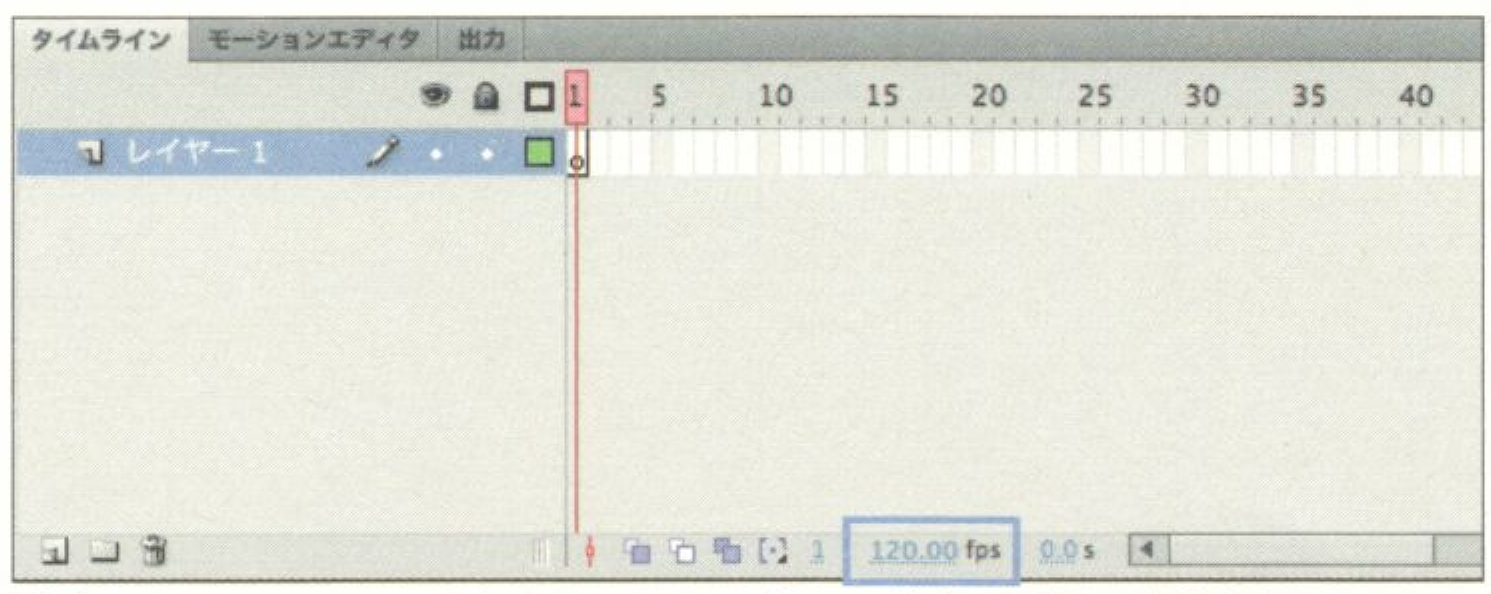
新規ファイルを作成したら、ステージサイズを設定します。[プロパティ] パネルの [プロパティ] をクリックし、[サイズ] 右の [編集] ボタンをクリックして [ドキュメントプロパティ] ダイアログから設定します。
フレームレートの設定は、[プロパティ] の [フレームレート] に数値を入力します。入力できる数値は [0.01] ～ [120] です。

プロパティパネル　　ドキュメントプロパティ



フレームレートの設定

タイムライン

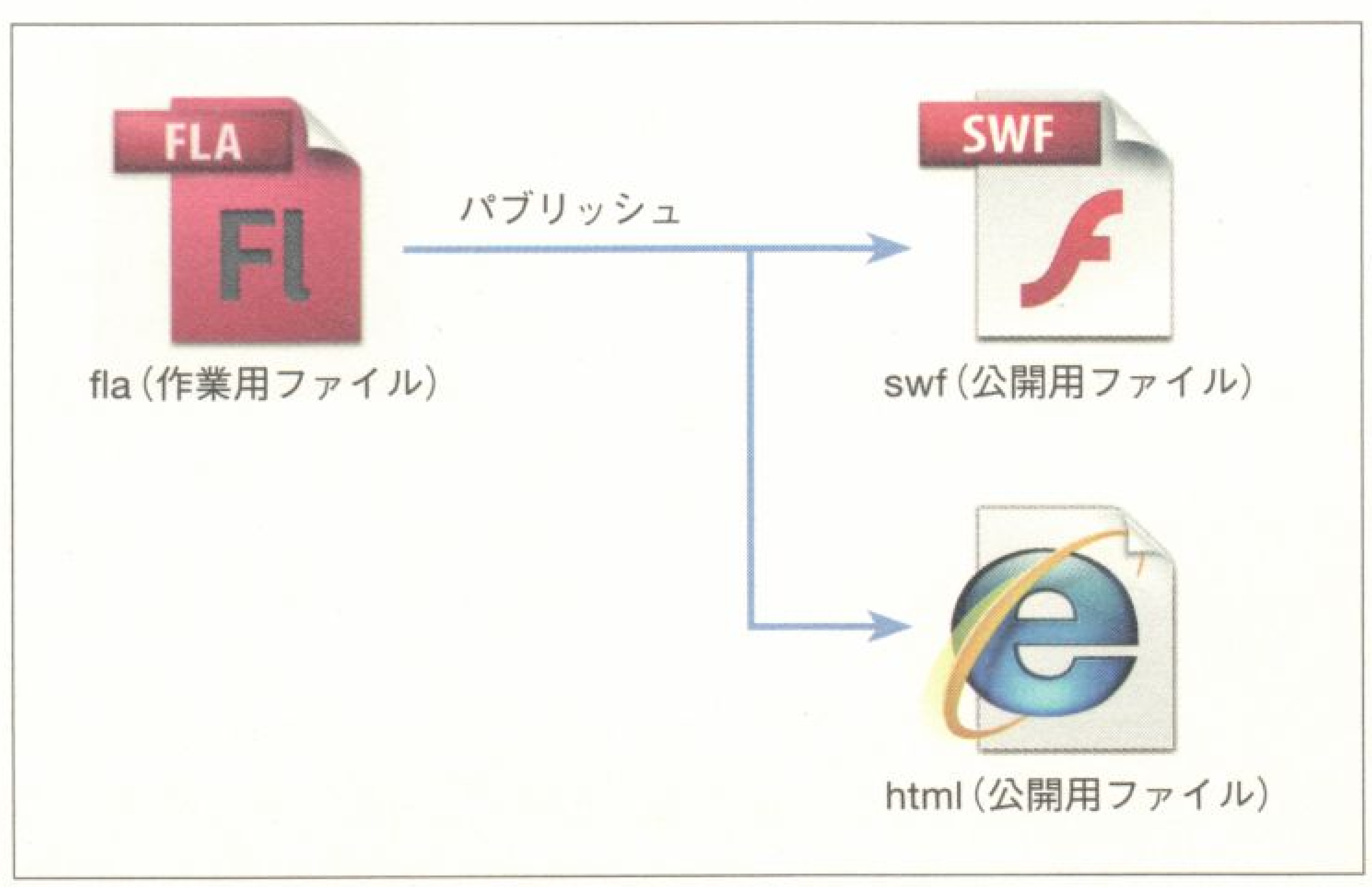


設定したフレームレートはタイムラインウィンドウの下部に表示される

パブリッシュ

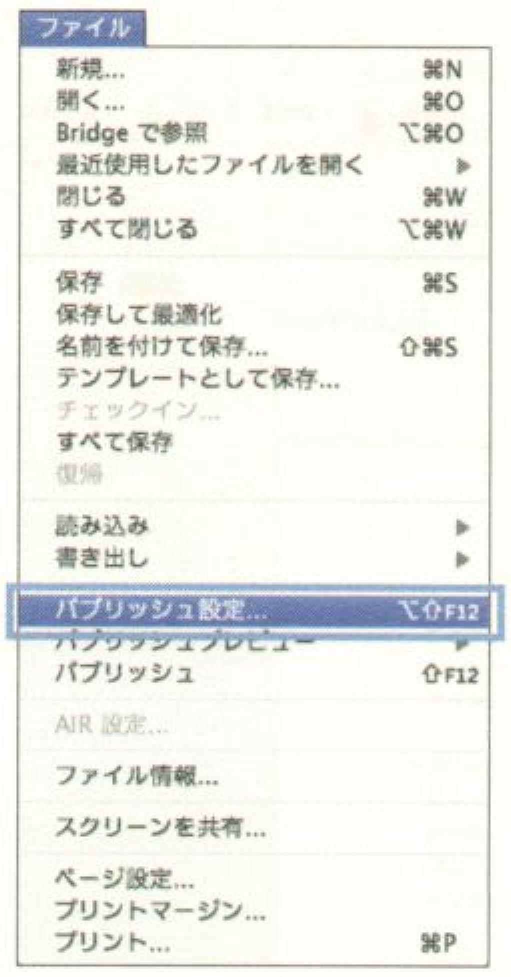
ファイルが完成したら、Flash ドキュメント(.fla)から再生用の Flash ムービー(.swf)や HTML ドキュメントなどを作成します。これをパブリッシュと言います。

Flash ドキュメントのままでは、Web上で再生できません。よって、作品が完成したら、ブラウザで再生できる形式(swf形式)に書き出す必要があります。完成する前でも、動作確認用に書き出すことがあります。また、パブリッシュによって、swf形式だけでなく、GIF、JPEG、PNG、QuickTime などの代替ファイル形式を書き出すこともできます。



▶▶ パブリッシュの設定

［ファイル→パブリッシュ設定］で、パブリッシュ設定画面を開きます。



「形式」タブ

書き出しの形式タイプを選択します。Flash（swf）以外にHTML（html）を作成したい場合は、「HTML」にもチェックを入れましょう。

「Flash」タブ

FlashPlayerのバージョンとスクリプトの設定を確認します。

「HTML」タブ

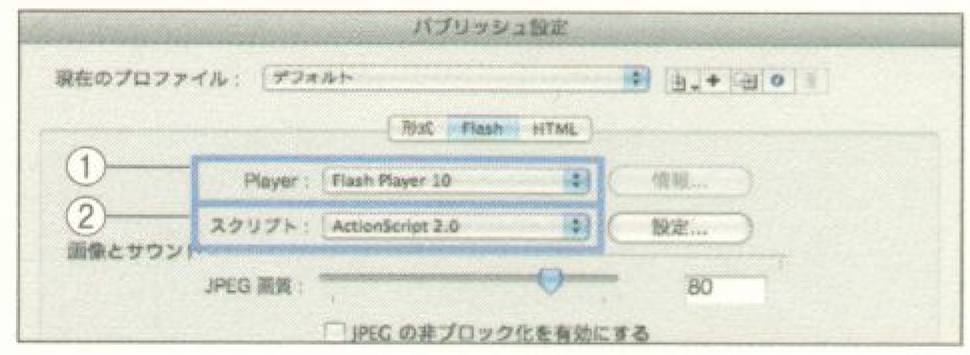
htmlファイルを書き出す場合の設定をします。ファイルをループさせて再生したい場合は「ループ」にチェックをします。

形式



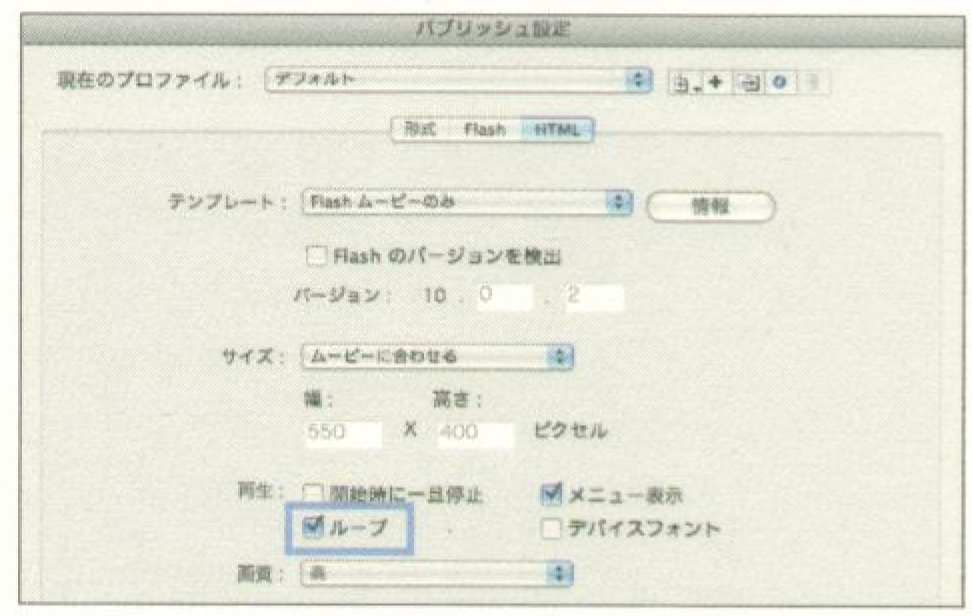
① 書き出したい形式にチェック
② ファイル名を選択

Flash



① FlashPlayerのバージョンを選択
② スクリプトタイプを選択

HTML

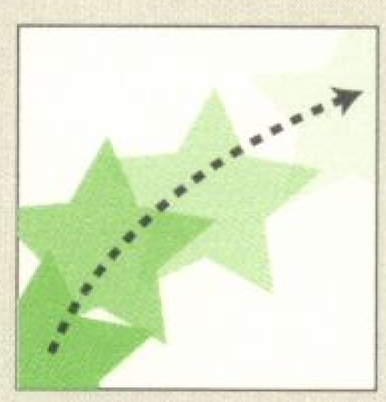


ループさせる場合にチェック

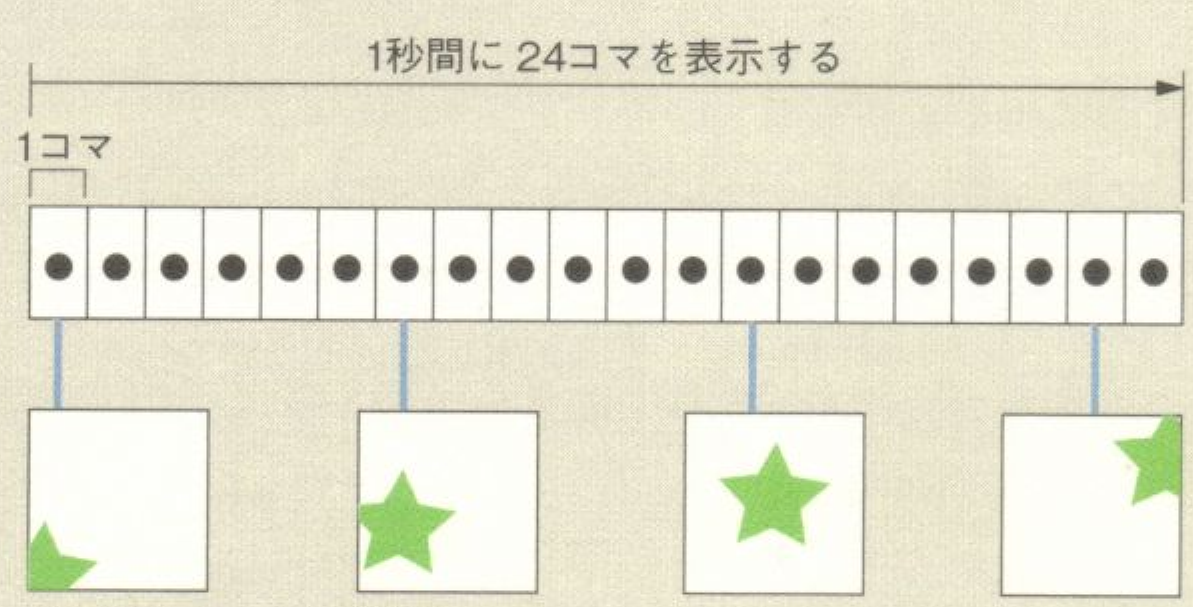
COLUMN

フレームレートとは

フレームレートとは、1秒間に再生されるフレームの数を示し、アニメーションの速度を決める際に使用されます。例えば、1秒間に24コマを表示するフレームレートの場合、1秒間で図のようにアニメーションが変化していきます。
フレームレート数が高くなればなるほど、1秒間に表示されるフレーム数も多くなり、より詳細なアニメーションの表現が可能です。その一方、1秒間に表示する描画数も比例するため、デバイスに負荷がかかります。
なお、フレームレートはFlash ムービー全体に適用され、さらにはActionScriptのEnterFrame関数にも適用されますので、注意が必要です。EnterFrame 関数は、このフレームレートで設定された数値の回数/1秒間に実行されます。



星型が斜め右上に移動するアニメーション



アニメーションの遷移

LECTURE 1-3

Flashの基本

ツールの使い方

ツールパネルのツールでは、オブジェクトを編集したり、描画したりすることができます。大まかに分けて、「描画関連」と「表示」と「カラー設定」と「オプション」の4つのツールがあります。

描画関連のツール

オブジェクトの描画やテキストの入力、カラーの塗りや変形についてのツール



【選択ツール】…………………… 選択範囲を指定します。

【ダイレクト選択ツール】…… パスのポイントやハンドルをダイレクトに選択できます。

【自由変形ツール】………… オブジェクトを拡大・回転するなど、自由に変形できます。

【3D回転ツール】……………… X、Y、Zの各軸、またはX軸とY軸の周りで自由に回転させることができます。

【なげなわツール】………… 自由な形で選択範囲を指定できます。

【ペンツール】………………… ベジェ曲線のラインを描けます。

【テキストツール】…………… テキストを入力します。

【線ツール】…………………… 直線が描けます。

【矩形ツール】………………… 楕円、四角形、扇形、多角形、星形などの図形を描けます。

【鉛筆ツール】………………… 自由に線を描くことができます。

【ブラシツール】…………… ブラシで色を塗ることができます。

【パターン描画ツール】…… あらかじめ用意された、つる模様などのパターンを簡単に描くことができます。

【ボーンツール】…………… アニメーションを作成する際に、関節のような動きを滑らかに表現することができます。

【バケツツール】…………… 線で囲まれた部分に色を流し込むことができます。また、一度塗った部分を新たな色で塗り直すことができます。

【スポイトツール】…………… 画面上の色を抽出することができます。

【消しゴムツール】………… 線、塗りを消すことができます。

カラー設定のツール

オブジェクトの塗りや線の設定を行うツール



【線のカラー】……… 線の色を表します。クリックするとパレットが表示され、好きな色を選択することができます。

【塗りのカラー】…… 塗りの色を表します。

【白黒】

【カラーの入れ替え】

表示のツール

表示比率の変更やスクロールを行うツール



【ハンドツール】……… ワークエリア内の表示させる位置を変更したり、スクロールバーをドラッグしたりできます。

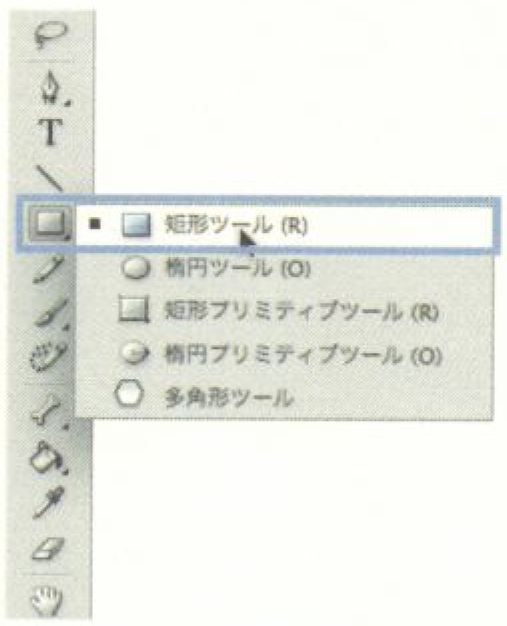
【虫めがねツール】…… ステージの画面表示を拡大・縮小します。

矩形ツールの使い方

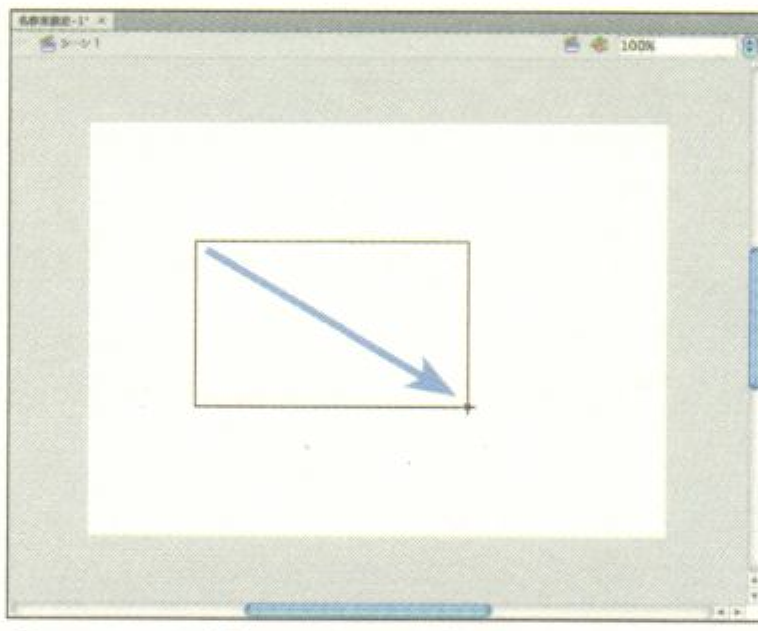
ツールの中でも使用頻度の高い、矩形ツールを使ってみましょう。
矩形ツールでは四角形のほか、多角形などを作成することもできます。

▶▶ 四角形の作成

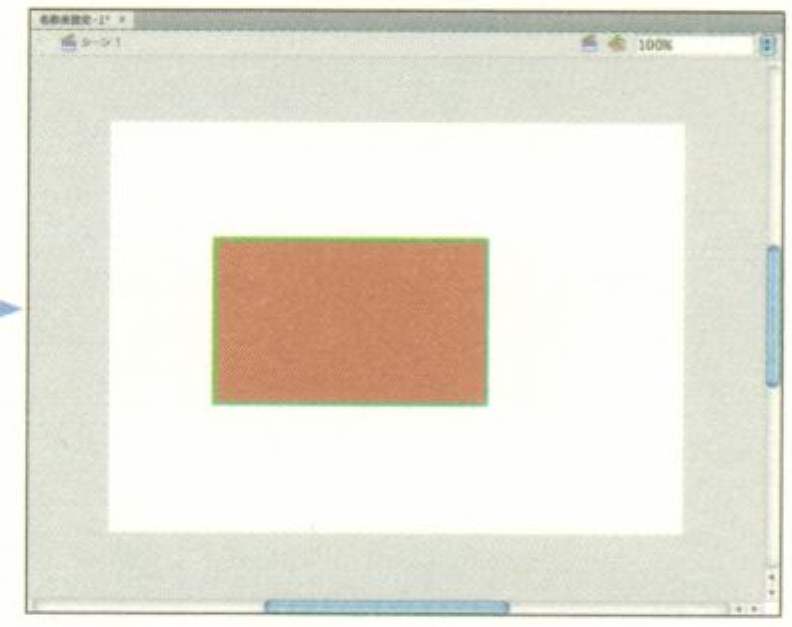
ツールパネルから、矩形ツールを選択し、ステージ上でドラッグします。



①矩形ツールを選択



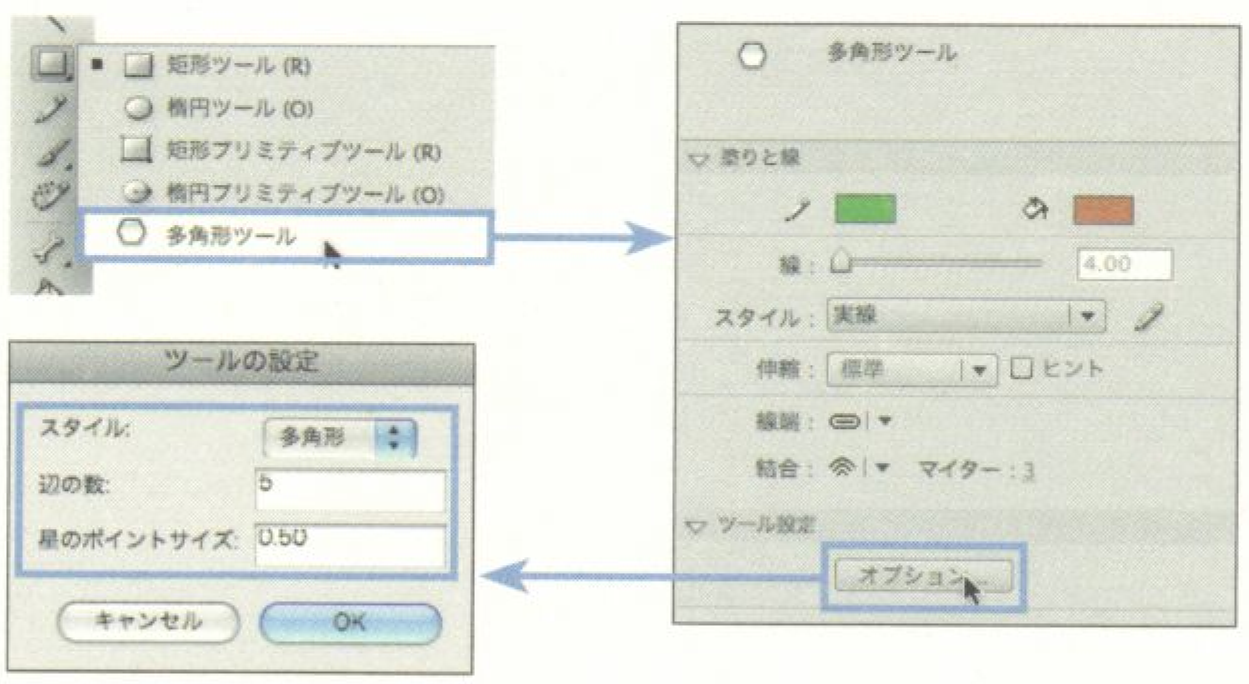
②ステージ上でドラッグ



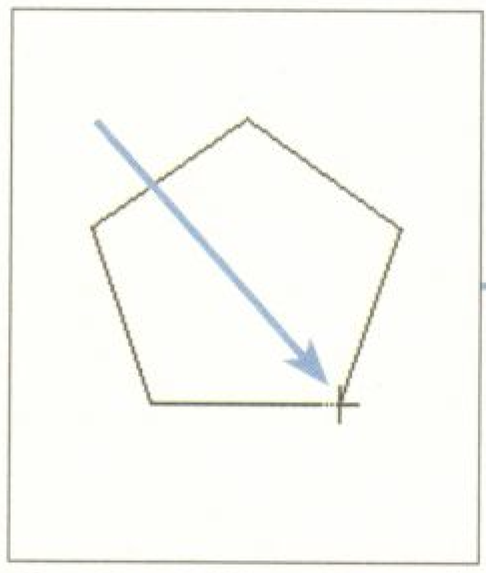
③矩形が作成される

▶▶ 多角形の作成

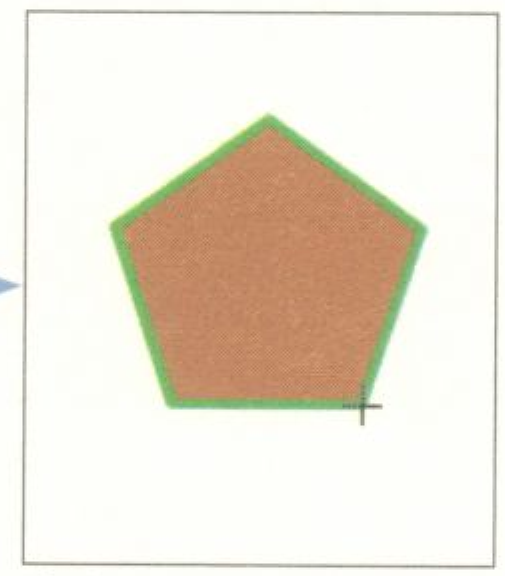
ツールパネルから、多角形ツールを選択します。プロパティの［オプション］から、多角形の設定が行えます。



①プロパティのオプションを選択し、多角形の設定を行う



②ステージ上でドラッグ

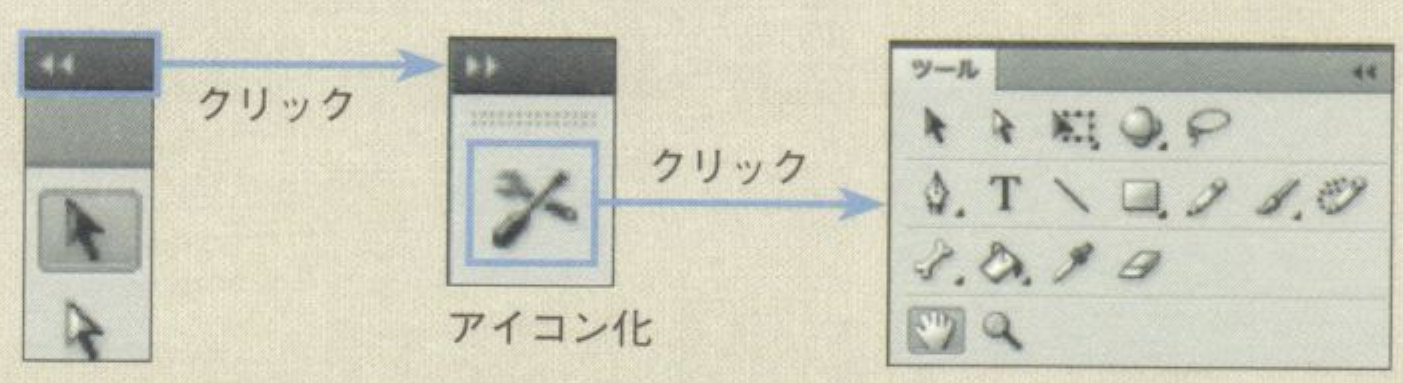


③多角形が作成される

COLUMN

ツールパネルの表示の切り替え

ツールパネルは、アイコン表示にしたり、
パネルとして表示させることができます。

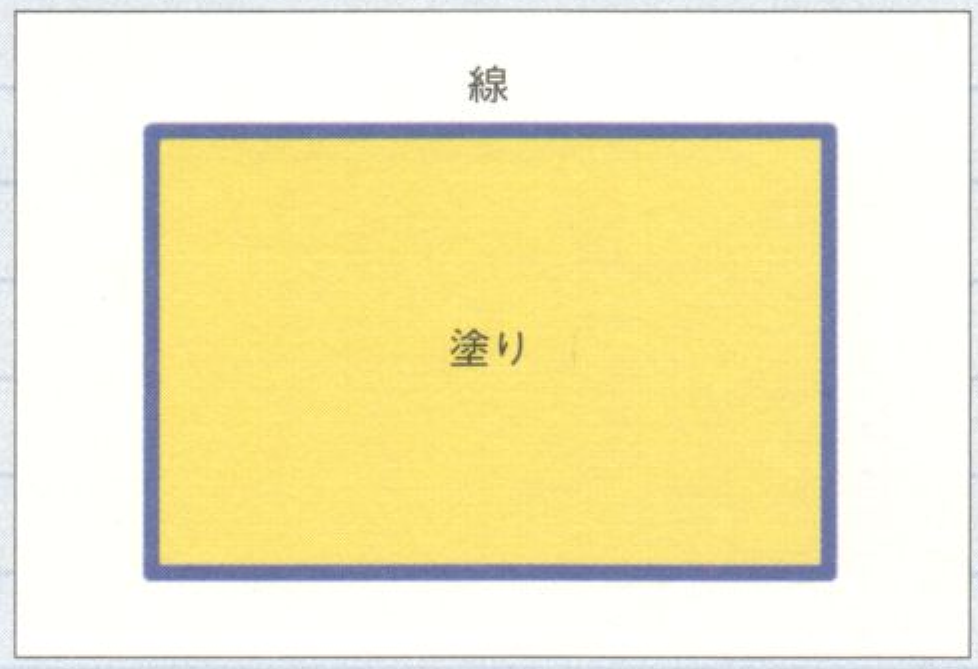


LECTURE 1-4

Flashの基本

線と塗り

矩形ツールで描画した楕円や四角形などのオブジェクトは、輪郭になる部分の「線」と内側になる部分の「塗り」で構成されています。Flashでは、絵を描く際に「線」と「塗り」は別々のものとして扱います。線と塗りを別々のものとして扱うので、「線のカラー」「塗りのカラー」を分けて設定できたり、線を分割することができます。

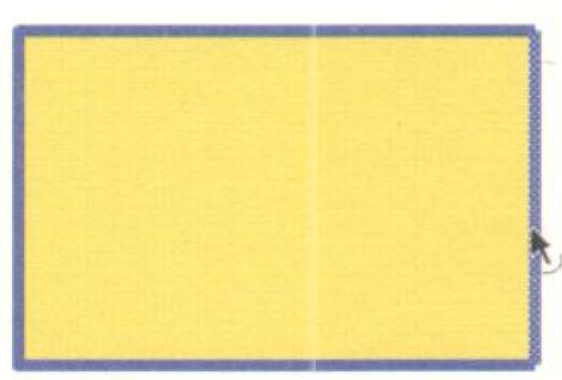


線と塗りの使い方

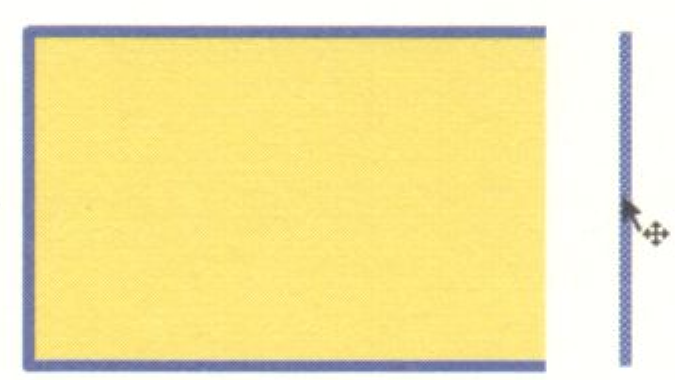
線となる部分と塗りとなる部分について、特徴的な箇所をまとめました。

▶▶ 線

線は塗りの外側の枠の部分になります。
線を分割するためには、線の一辺を選択し、ドラッグして分離します。



①選択ツールで線の一辺を選択

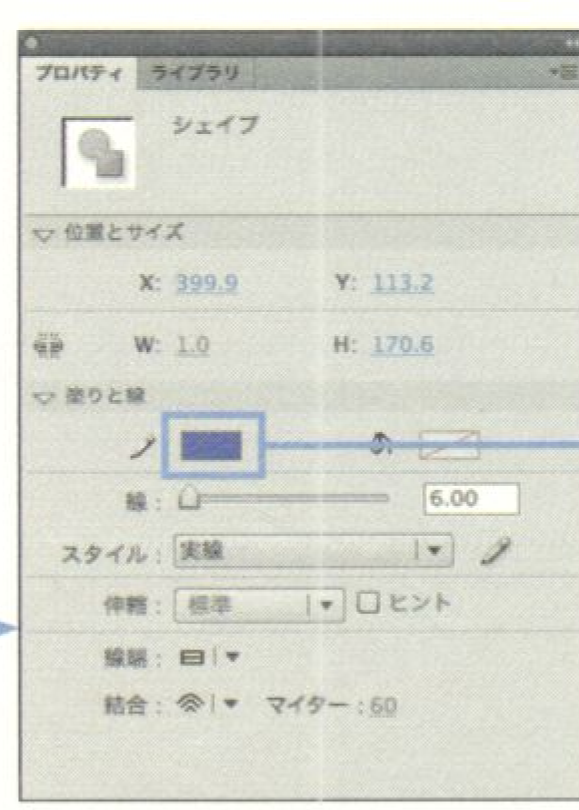


②ドラッグして移動

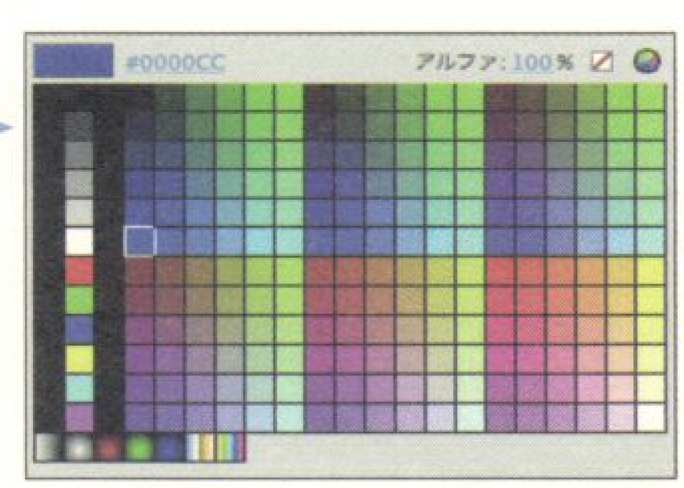
線全体の色を変更するには、線をダブルクリックし、線を全選択した状態で、プロパティパネルの[塗りと線]で色を変更します。



①選択ツールで線をダブルクリックする



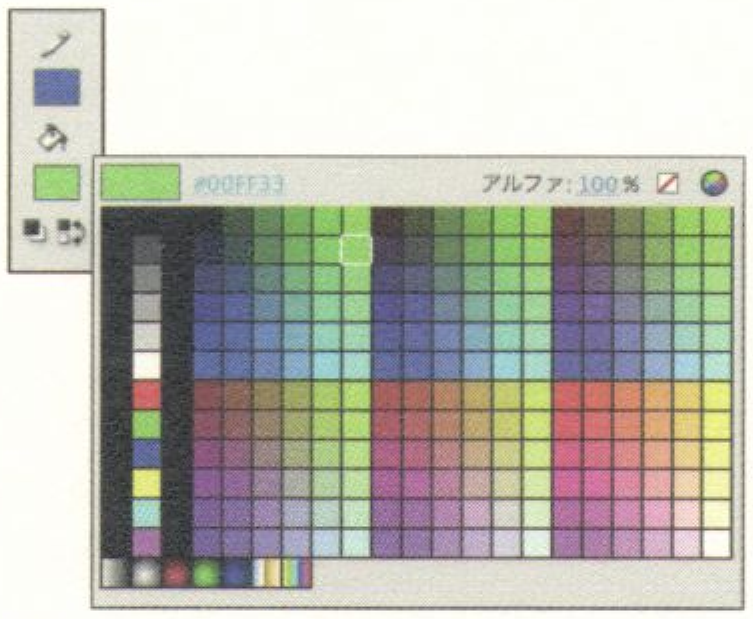
②クリック



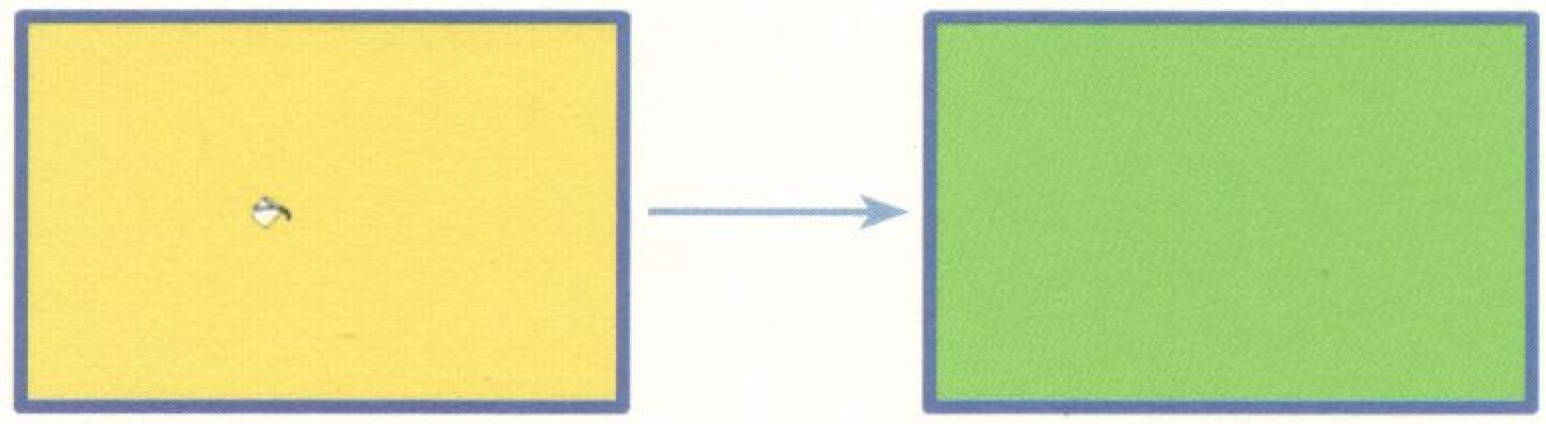
③線の色を選択

▶▶ 塗り

塗りは、図形の内側部分になります。
バケツツールで色を塗る場合、塗りたい色を「塗りのカラー」に選択した状態で、バケツツールで塗りに適用します。

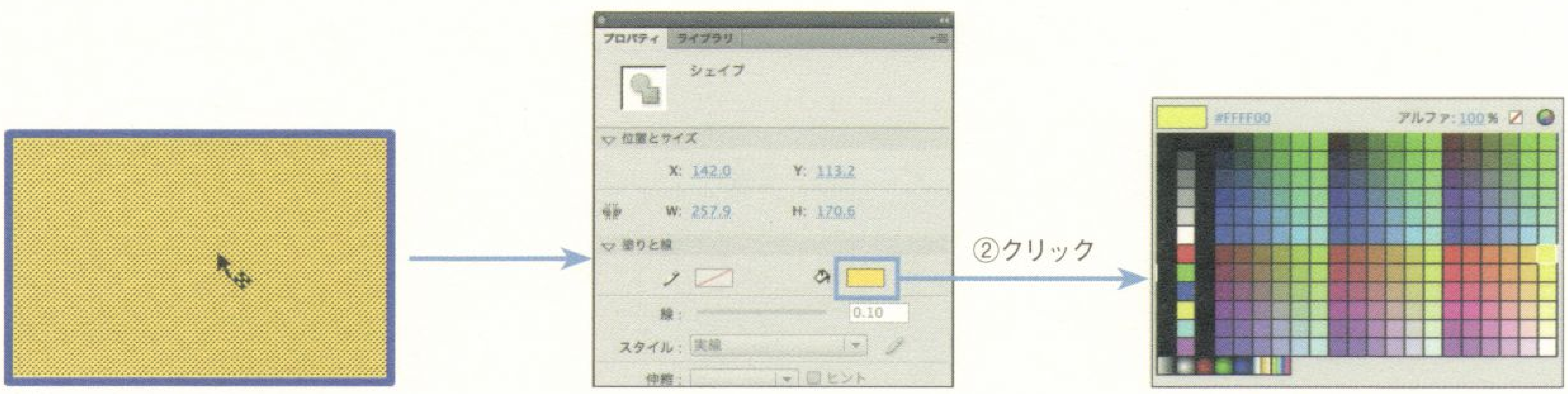


①バケツツールを選択し、色を変更する



②塗りに適用

選択ツールで色を塗る場合、塗りの部分を選択した状態で色を変更します。

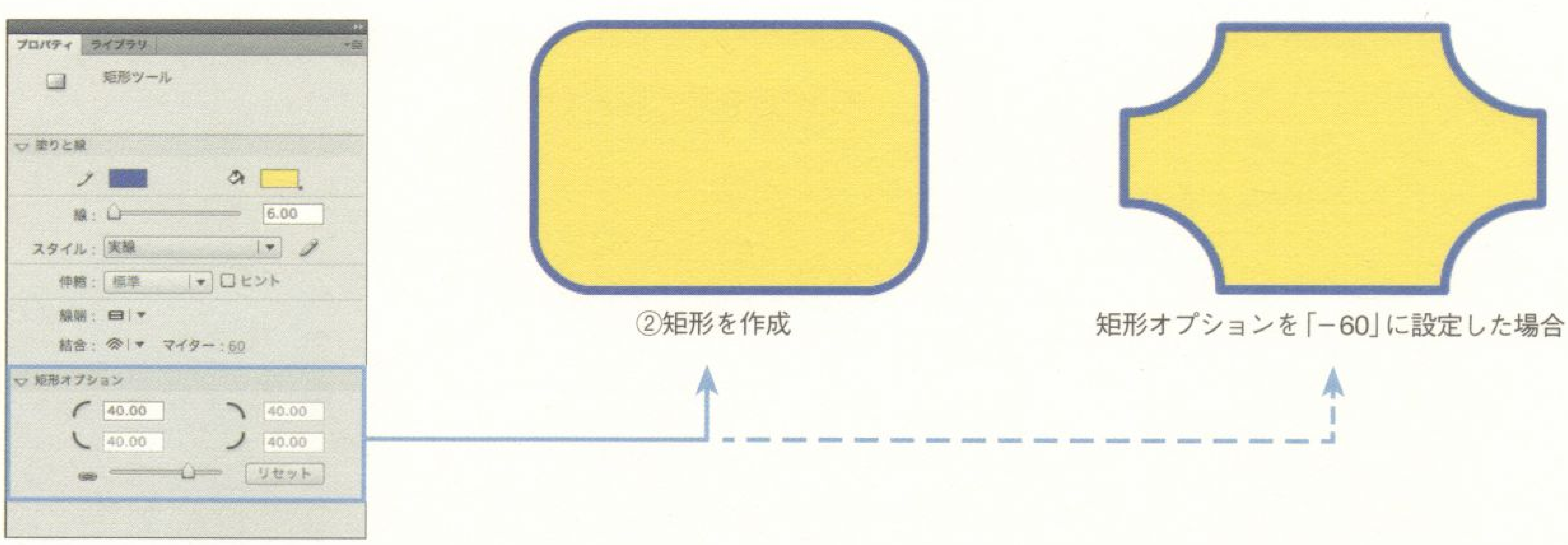


①塗りの部分を選択

③塗りの色を選択

▶▶ 角丸四角形を作成するには

矩形ツールで矩形を描く際に、プロパティパネルを使って、線の部分の形を変形することができます。矩形ツールを選択し、プロパティパネルの [矩形オプション] で角丸の半径を設定してから、矩形を描きます。角丸の半径を「40」に設定した場合は、図のような角丸四角形を描けます。



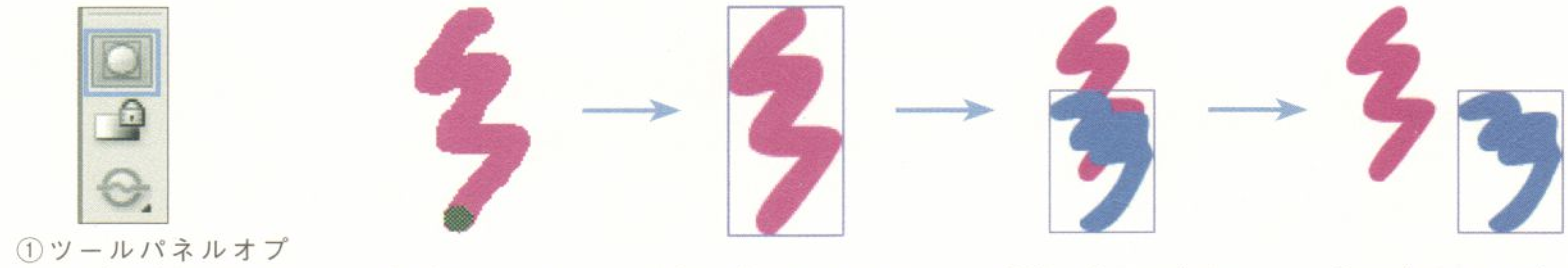
②矩形を作成

矩形オプションを「−60」に設定した場合

①矩形オプションを「40」に設定

▶▶ オブジェクトの描画オプションについて

矩形ツールのほか、ペンツール、線ツール、鉛筆ツール、ブラシツールで、オブジェクトの描画オプションを使用すると、他の描画オブジェクトと交わらない図形が描画されます。この状態のときに描かれた線や塗りは、描いた時点で描画オブジェクト化されるので、重なりを気にせずにどんどん描けるのでとても便利です。



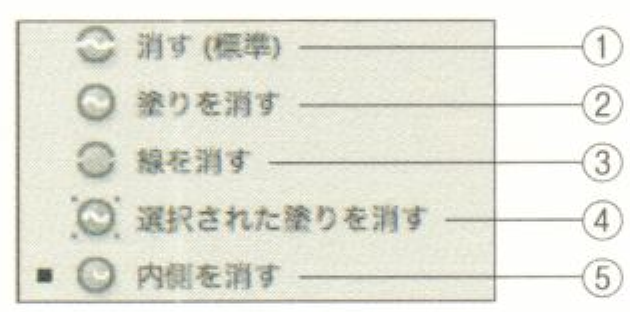
①ツールパネルオプション［オブジェクトの描画］をONにする　②ブラシツールなどを使用する　③オブジェクト化された状態で描かれる　④他の描画オブジェクトを重ねて描く　⑤オブジェクトごとに分離できる

消しゴムツール

描いた塗りや線を消すには、消しゴムツールを使います。消しゴムツールでは、クリックした範囲にある線や塗りだけを消去したり、消しゴムをドラッグした位置にある図形を消去するなど、さまざまな消去方法を選択できます。

消しゴムモード

ドラッグした部分の塗りだけを消去、線だけを消去、選択部分の塗りだけを消去など、特殊な消去ができます。

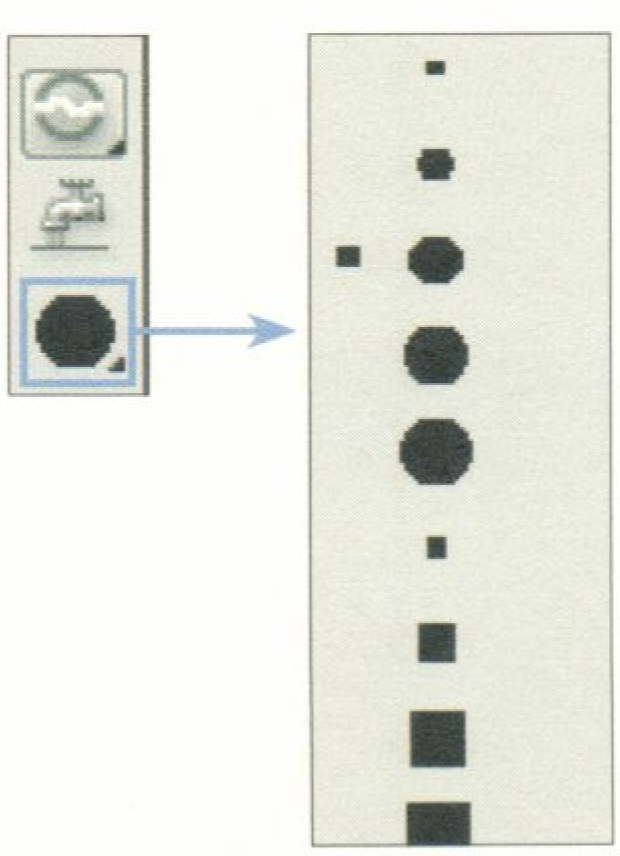




消去する図形　①ドラッグした部分を消去　②ドラッグした部分の塗りだけを消去　③ドラッグした部分の線だけを消去　④選択した塗りのドラッグした部分を消去　⑤ドラッグの始点にある塗りのみドラッグした範囲を消去

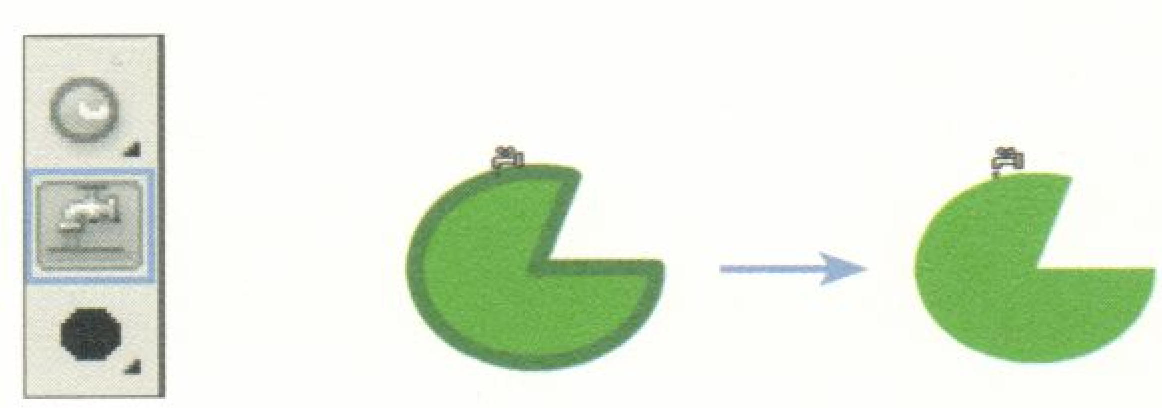
消しゴムの大きさと形

消しゴムの大きさと形を選択することができます。



流し消す

［流し消す］をオンにした状態で流し消したい線や塗りにカーソルを合わせてクリックすると、クリックした位置にあった線や塗りが消去されます。



①ツールパネルオプション［流し消す］をON　②消去したい部分にカーソルの水滴を合わせる　③クリックした位置にあった線が消去される

LECTURE 1-5

Flashの基本

シンボル

シンボルとはFlashで作成されたムービー内で動くキャラクターやボタンなどを指します。
Flashでアニメーションさせるには、個々の要素がシンボルに変換されている必要があります。
シンボルは3種類あり、それぞれの用途によって使い分けます。

シンボルとライブラリ

シンボルは3種類あります。
作成したシンボルは［ライブラリ］に格納され、タイプはあとから変更することもできます。

▶▶ シンボルの種類

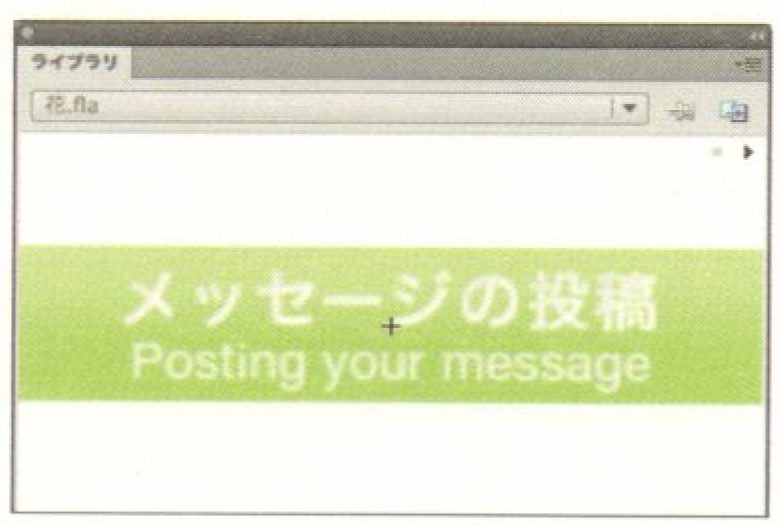
- ・ムービークリップシンボル …… ActionScriptで扱うことができる
- ・ボタンシンボル ………………… ロールオーバーの演出などを行う（AS 3ではボタンの定義ができない）
- ・グラフィックシンボル ………… 基本はアニメーションのみのときに使う

［ライブラリ］パネルに格納されたシンボルのアイコンを見れば、そのシンボルのタイプが分かります。

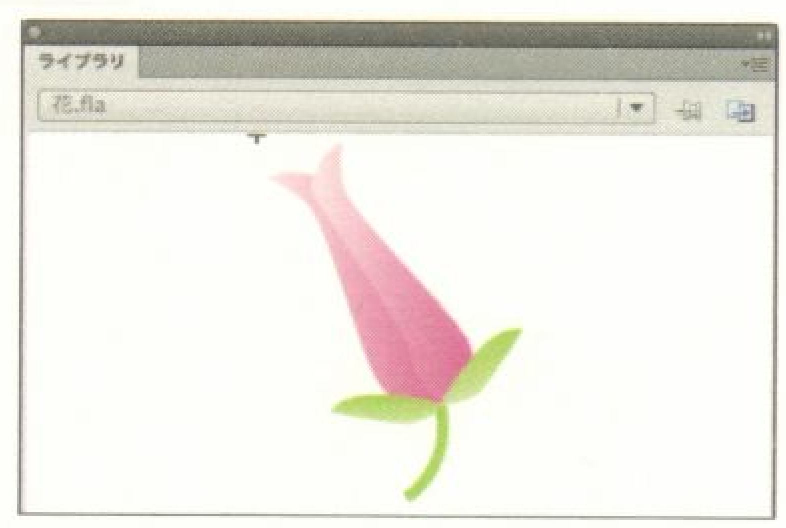
ムービークリップシンボル



ボタンシンボル



グラフィックシンボル



▶▶ シンボルの作成

シンボルを作成する方法は2つあります。

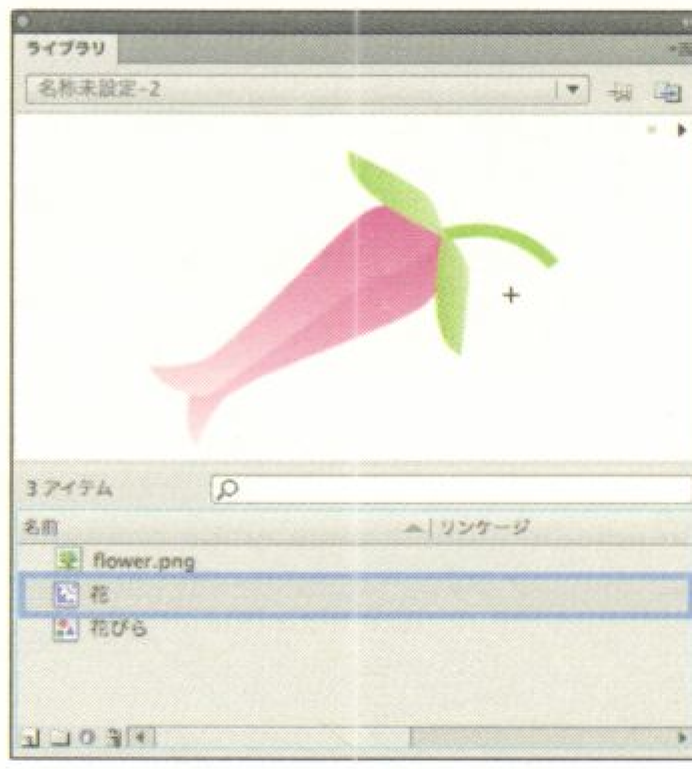
- ・［挿入→新規シンボル］を選択し、［新規シンボルの作成］ダイアログ（右図）で名前や種類を選択します。

- ・先にシンボル用のグラフィックを描き、それを選択した状態で［修正→シンボルに変換］を選びます。［シンボルに変換］ダイアログが表示されるので、名前や種類を設定します。

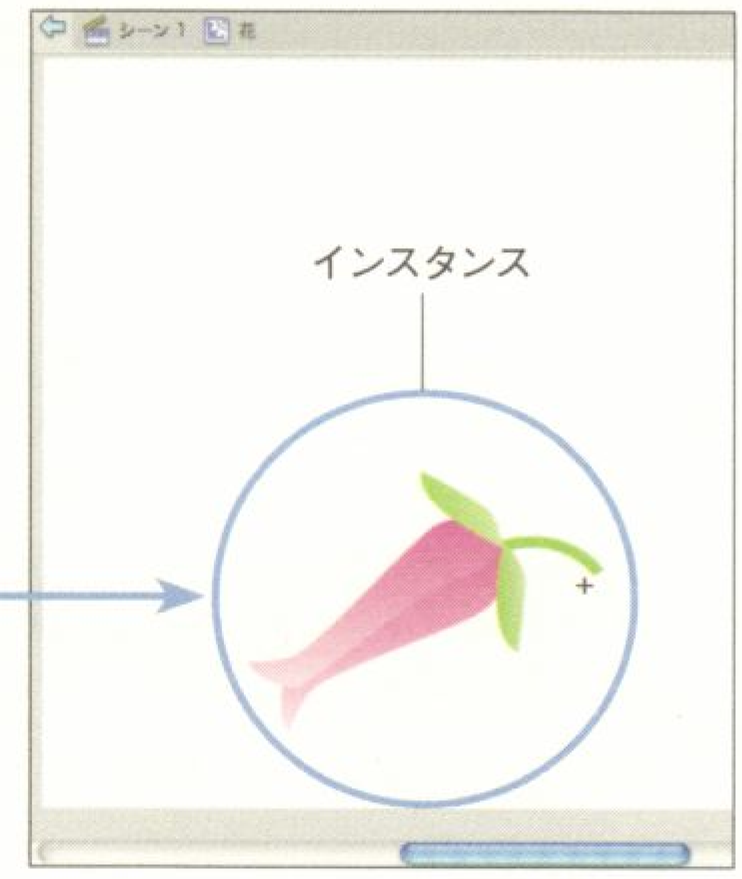


▶▶ ライブラリについて

ライブラリには、Flashファイル内に含まれるすべてのシンボルとアセットが保存されています。シンボルなどは作成したときに自動的にライブラリに追加されます。 ここで、すべてのシンボルの管理をしています。
シンボルを配置したい場合は、[ライブラリ]パネルを開き、シンボルを選択して、ステージ上にドラッグします。



ステージ上にドラッグして配置



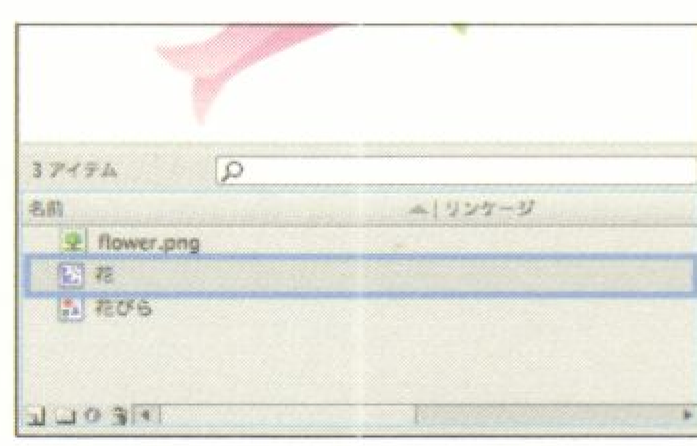
ライブラリからステージ上に配置されたオブジェクトは、「インスタンス」と呼ばれます。インスタンスはシンボルのコピーです。Flashではシンボルを作成した後、それをインスタンスとしてワークスペース上に何個も配置することができます。同一形状のオブジェクトの場合、それぞれをシンボルに変換して使用するよりも、インスタンスとして配置する方がファイルのサイズが軽くなります。

シンボルの使い方

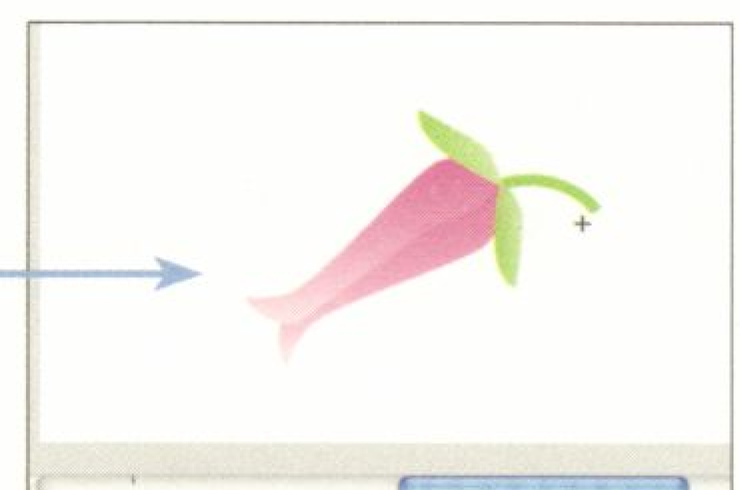
それでは具体的に、それぞれの種類のシンボルについて、
特徴となる部分を説明していきましょう。

▶▶ ムービークリップシンボル

シンボル独自のタイムラインを持ち、アニメーション本体のフレーム数に関係なく再生されます。メインのタイムラインの1つのフレームのみにムービークリップを配置するだけで、アニメーションを再生することができます。モーショントゥイーンを作成するほか、ActionScriptを使って動作をコントロールするときに利用します。動作を確認するには、ムービープレビューを実行します。



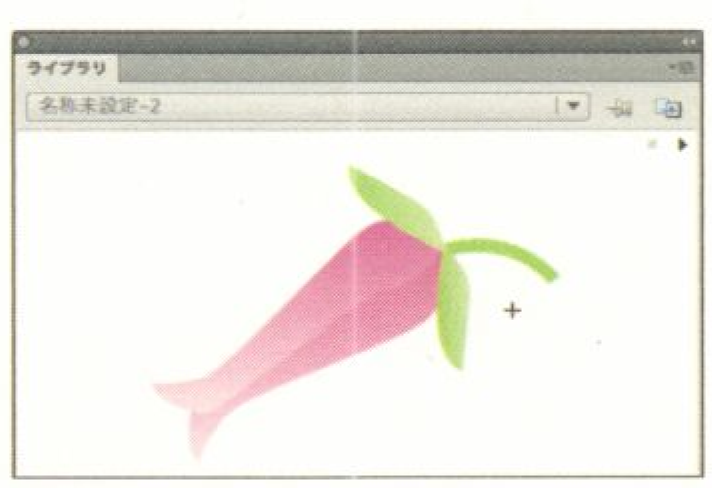
花（ムービークリップシンボル）



ムービークリップの中身



ムービークリップにはインスタンス名を付けることができます。



①ムービークリップを選択



②プロパティでインスタンス名を作成

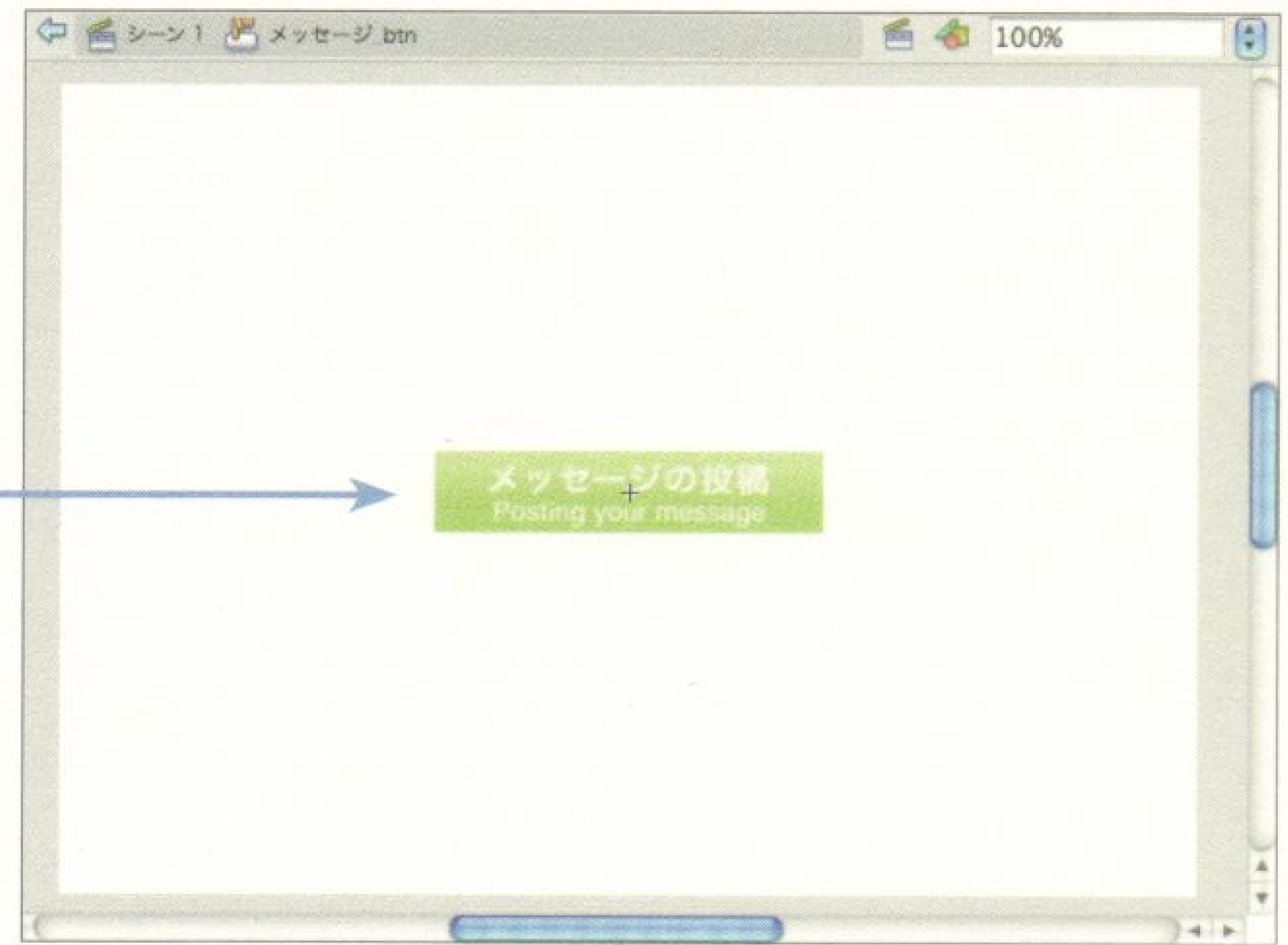
▶▶ ボタンシンボル

マウスオーバーやクリックなどのマウス操作ができるボタンを作成するときに利用します。マウスの動きに応じた「アップ」「オーバー」「ダウン」「ヒット」の4つのフレームが表示され、それぞれのオブジェクトを配置することができます。



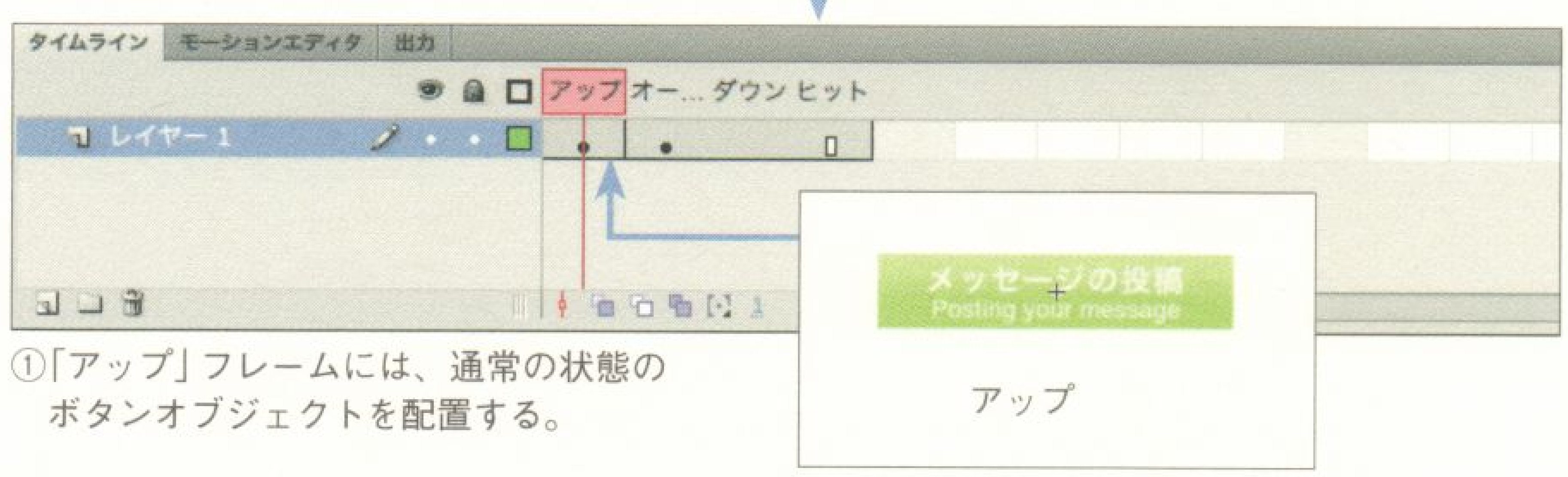
メッセージ_btn（ボタンシンボル）

配置

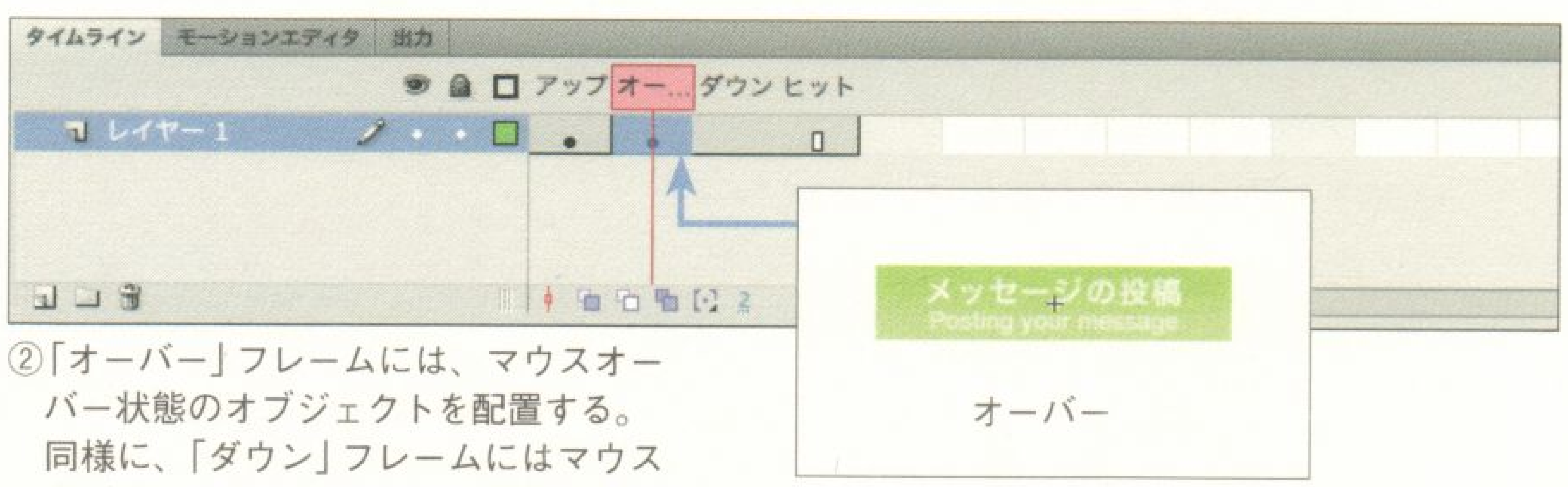


ボタンの中身

シンボルの中は4つのフレームがある



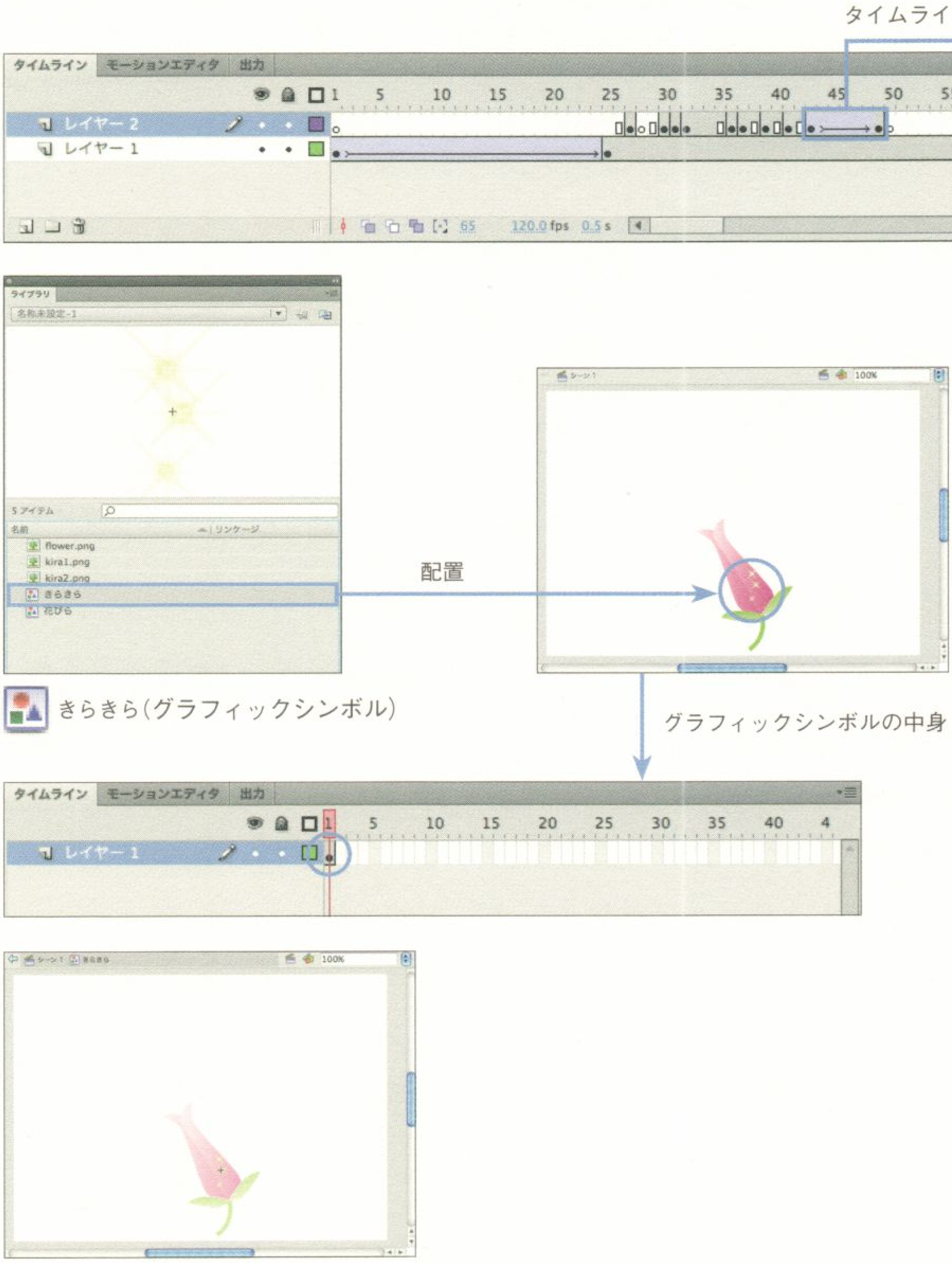
①「アップ」フレームには、通常の状態のボタンオブジェクトを配置する。

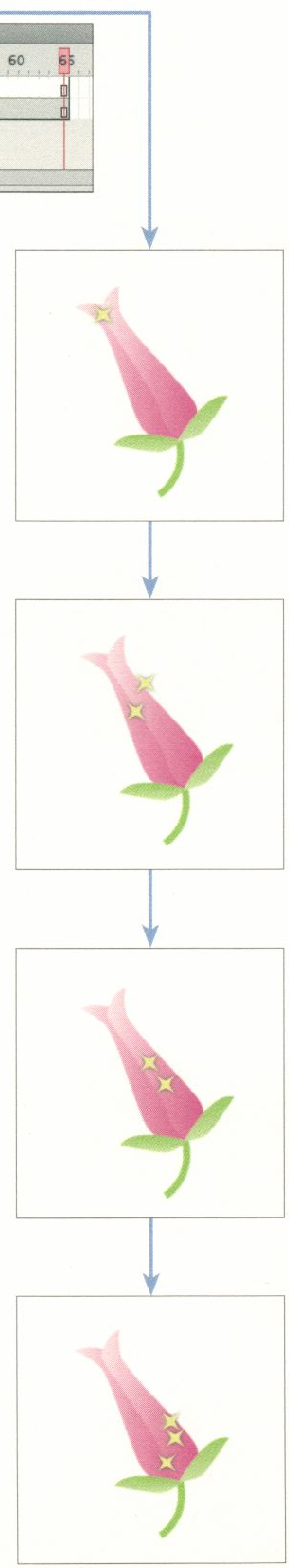


②「オーバー」フレームには、マウスオーバー状態のオブジェクトを配置する。
同様に、「ダウン」フレームにはマウスをボタン上でクリックした状態のオブジェクトを配置する。
「ヒット」にはボタンとして認識する領域を指定する。

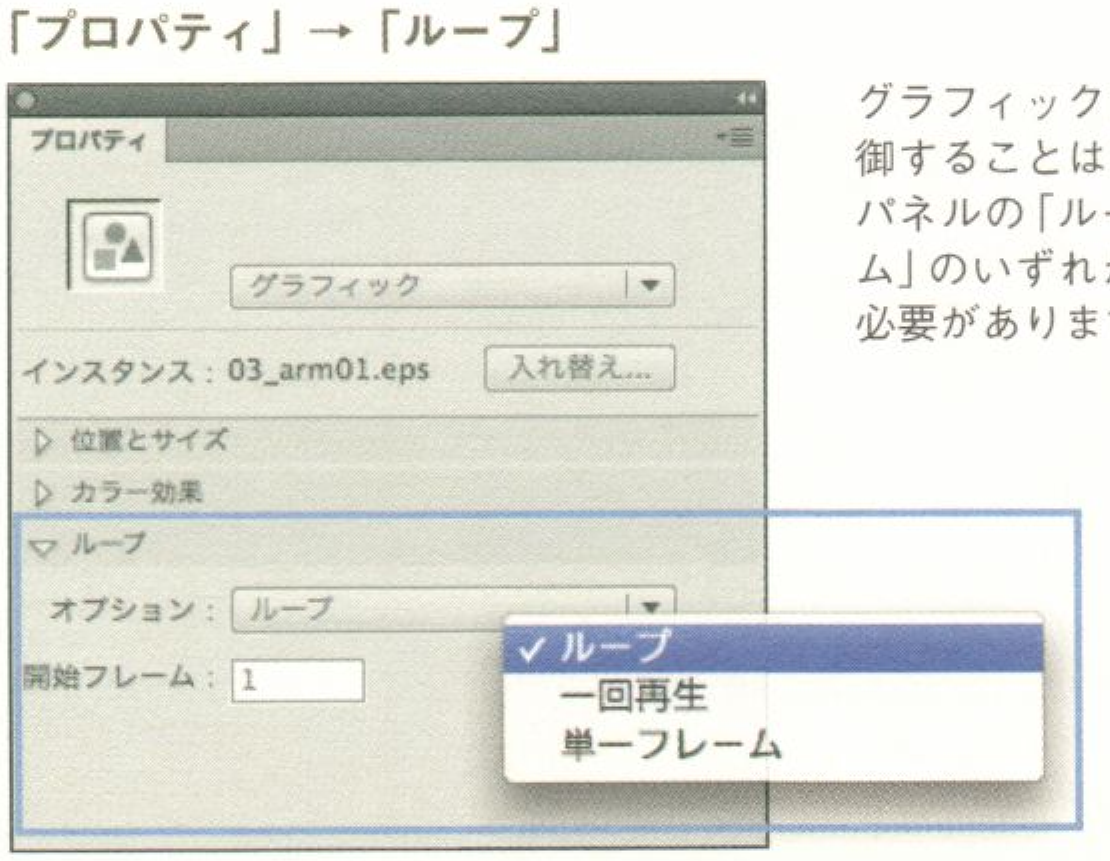
▶▶ グラフィックシンボル

ステージで使う図形や文字、ビットマップなどの静止したグラフィックを1つのパーツとして登録したシンボルです。タイムライン上でそれぞれのシンボルを動かしてアニメーションをさせることができます。





「プロパティ」→「ループ」



グラフィックシンボルはActionScriptで制御することはできないので、［プロパティ］パネルの「ループ」「一回再生」「単一フレーム」のいずれかから、再生方法を選択する必要があります。

LECTURE 1-6

Flashの基本

フレームとタイムライン

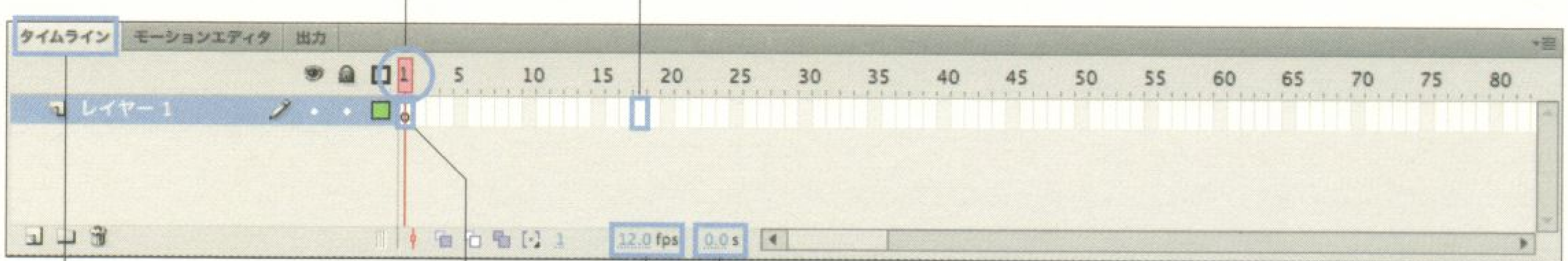
タイムラインは、Flashアニメーションの「時間軸」になる部分です。各マス目上の「フレーム」に配置したオブジェクトは、時間軸に沿って再生されます。

タイムラインパネルについて

[タイムライン] パネルはこのような構成になっています。

[再生ヘッド] 選択中のフレームを示す

[フレーム] アニメーションの1コマを表す



[経過時間] 再生ヘッドがあるフレームまでの再生時間を表示する

[フレームレート] 1秒あたりのアニメーションが再生されるフレーム数

[カレントフレーム] 再生ヘッドがあるフレーム

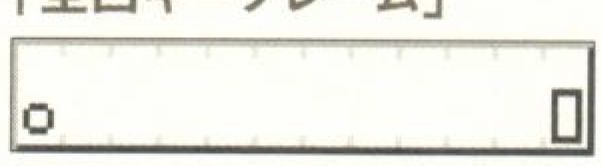


[タイムライン] の文字をダブルクリックすることで表示を折り畳むことができる

▶▶ フレームの種類

オブジェクトの有無や、設定されているアニメーションの種類など、状況によって、フレームの表示内容が変わります。

「空白キーフレーム」

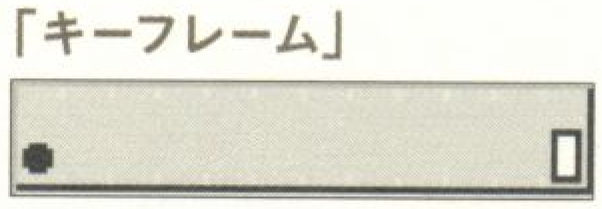


何も入っていないキーフレーム



ActionScriptが記述されたフレーム

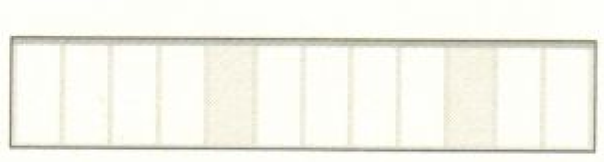
「キーフレーム」



図形やテキストなどのオブジェクトが配置されているキーフレーム



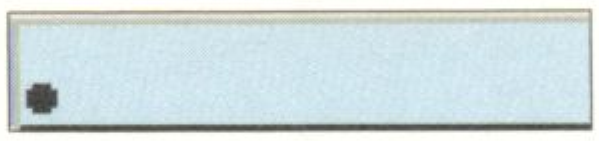
サウンドが設定されたフレーム



なにも設定されていないフレーム



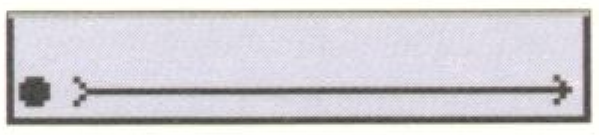
フレームラベルが設定されたフレーム



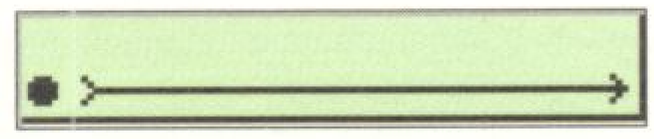
モーショントゥイーンが設定されたフレーム



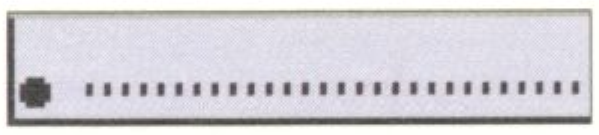
コメントラベルが設定されたフレーム



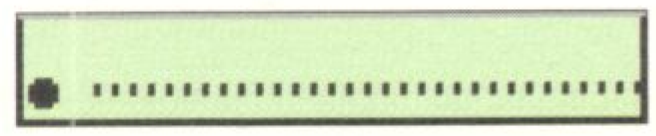
クラシックトゥイーンが設定されたフレーム



シェイプトゥイーンが設定されたフレーム



クラシックトゥイーンは設定されているが、最後のキーフレームが設定されていないフレーム



シェイプトゥイーンが設定されているが、最後のキーフレームが設定されていないフレーム

フレームの追加と削除

タイムライン上で、フレームを挿入することでオブジェクトを表示させたり、表示させる時間やタイミングを指定します。

フレームの挿入

タイムライン上で右クリックをして、[フレームの挿入]を選択します。

フレームの削除

削除したいフレームで右クリックをして、[フレームを削除]を選択します。

キーフレームの挿入

タイムライン上で右クリックをして、[キーフレームの挿入]を選択します。

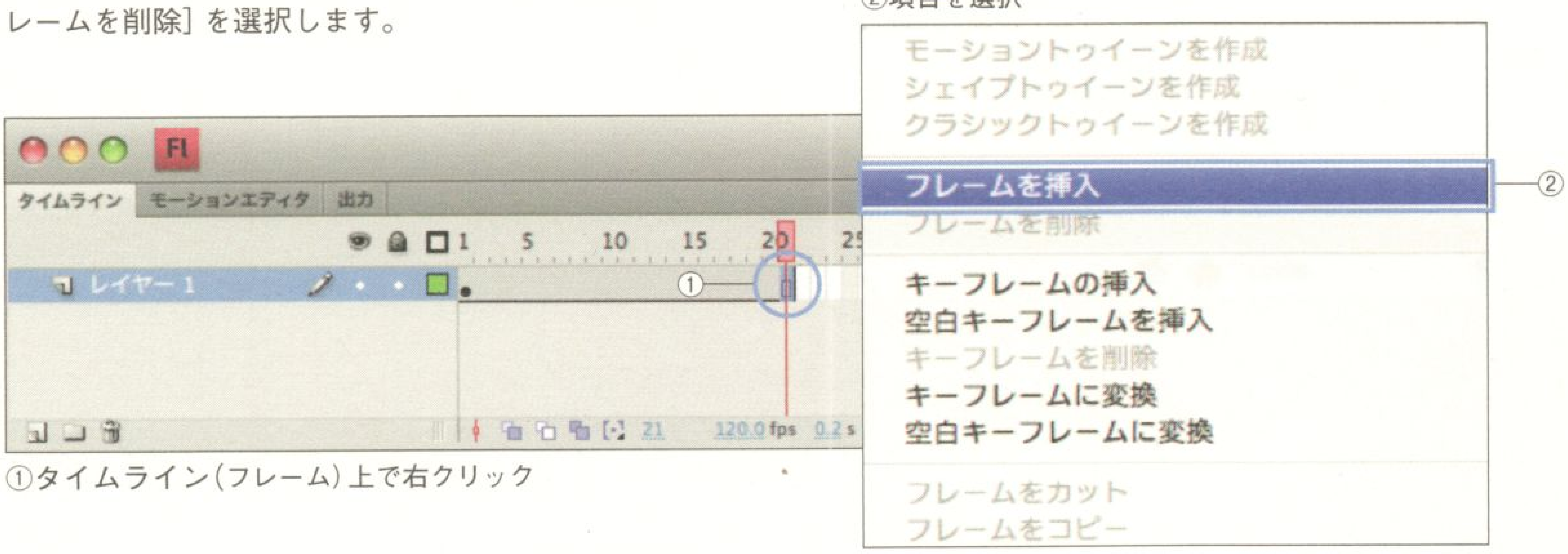
空白キーフレームの挿入

タイムライン上で右クリックをして、[空白キーフレームの挿入]を選択します。

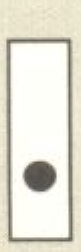
空白キーフレームの削除

削除したいフレームを右クリックして、[空白フレームを削除]を選択します。

②項目を選択

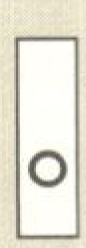


①タイムライン(フレーム)上で右クリック



オブジェクトがあるフレーム

選択中のレイヤーにオブジェクトがあると開始フレームに●が表示されます。これは「キーフレーム」と言い、オブジェクトがあることを示しています。



オブジェクトがないフレーム

レイヤーにオブジェクトがない場合は、フレームが○で表示されます。この○は「空白キーフレーム」と言います。

複数フレームの表示

通常、タイムライン上では再生ヘッドにあるフレームしか表示されませんが、
オニオンマーカーで選択された範囲内のタイムラインフレームの表示を見ることができます。

▶▶ オニオンスキン

オニオンスキン

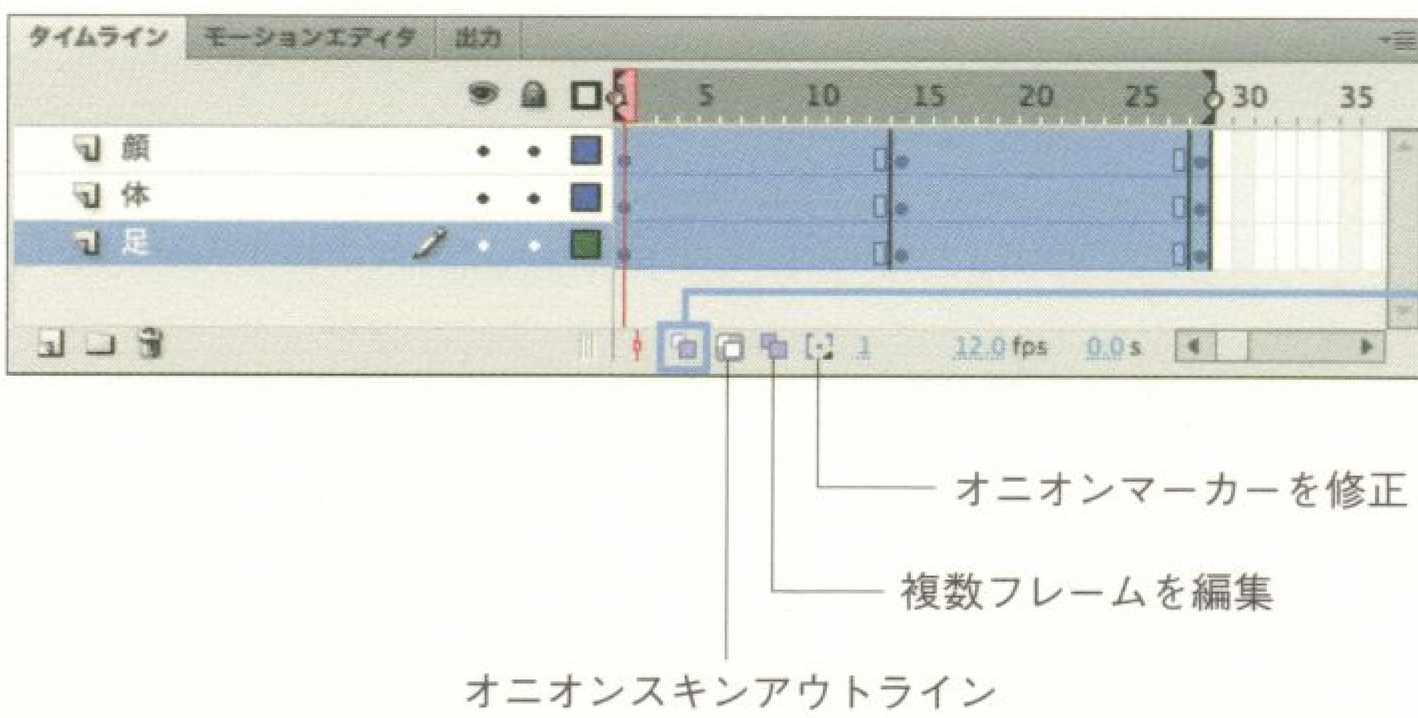
再生ヘッドの前後のフレームが透過して表示されます。表示される範囲は、[オニオンマーカーを修正] で設定します。

オニオンスキンアウトライン

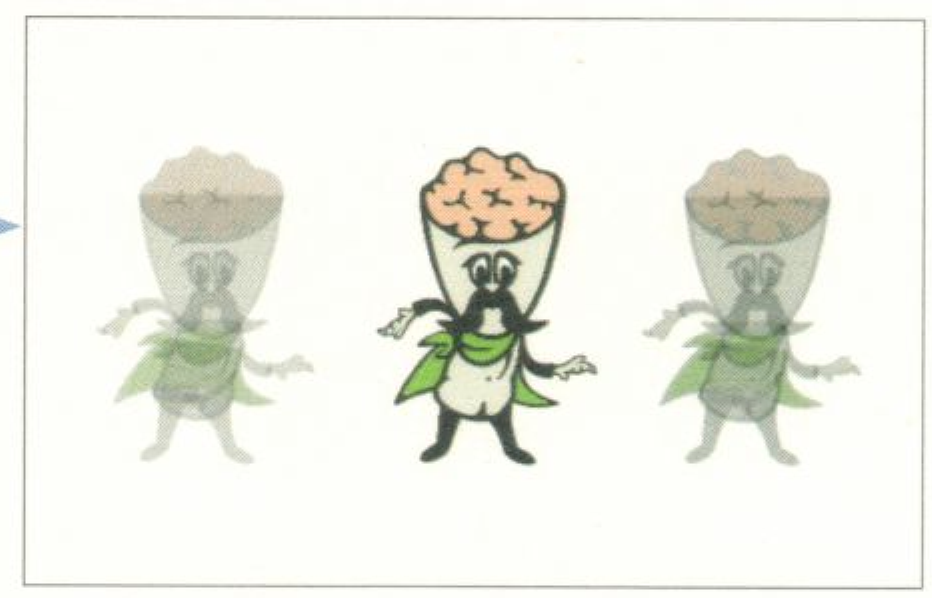
再生ヘッドのフレームとその前後のフレームにあるオブジェクトがアウトラインで表示されます。

複数フレームを編集

複数のフレームを選択してレイヤーの全体を移動する場合などは、「複数フレームを編集」を選択します。

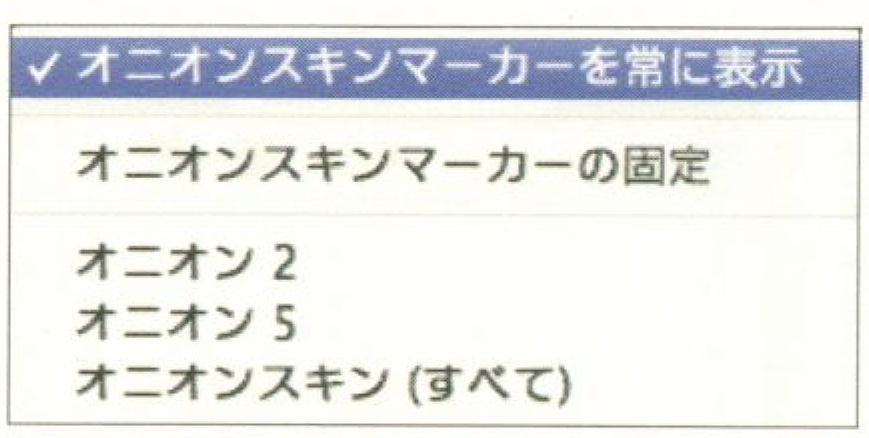


オニオンスキン



▶▶ オニオンマーカーを修正

[オニオンマーカーを修正] で選択する範囲を変更することができます。



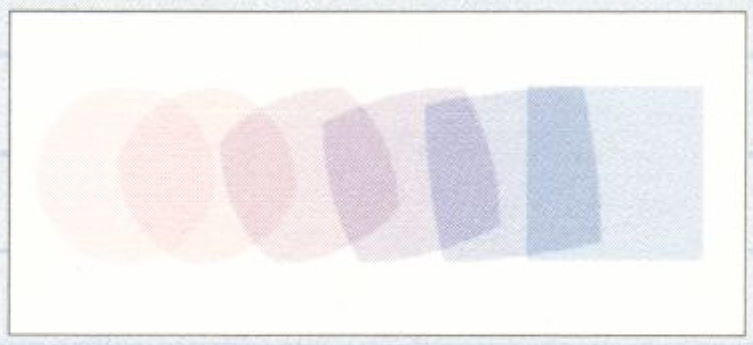
オニオンスキンマーカーを常に表示	オニオンスキン表示がオフのときでもオニオンスキンマーカーを表示します。
オニオンスキンマーカーの設定	再生ヘッドを動かしても、オニオンスキンマーカーの位置は固定されたままになります。
オニオン2	再生ヘッドの前後2フレームを表示します。
オニオン5	再生ヘッドの前後5フレームを表示します。
オニオンスキン(すべて)	すべてのフレームを表示します。

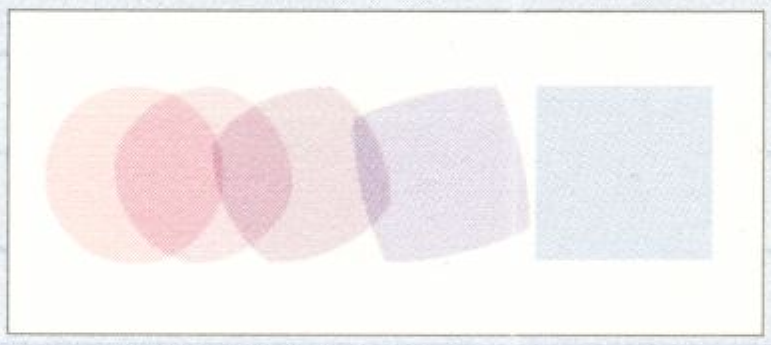
LECTURE 1-7

Flashの基本

イージング

イージングとは、モーショントゥイーンやシェイプトゥイーンの動きに変化を付ける機能です。次第に加速していく「イーズイン」と、次第に減速していく「イーズアウト」の設定ができます。トゥイーンの初期設定は、等速移動の動きですので、どうしても機械的な動きになってしまうため、滑らかな動きを加えることができるイージングは、Flashアニメーションでは頻繁に利用します。


設定なし：等間隔で移動


イーズイン：加速する


イーズアウト：減速する

イージングが使われているサンプル

■ PRACTICE 1：
コマ全体のアニメーションを作る … P180へ

合わせて読むと便利な項目

■ アニメーションを作る……………… P046へ

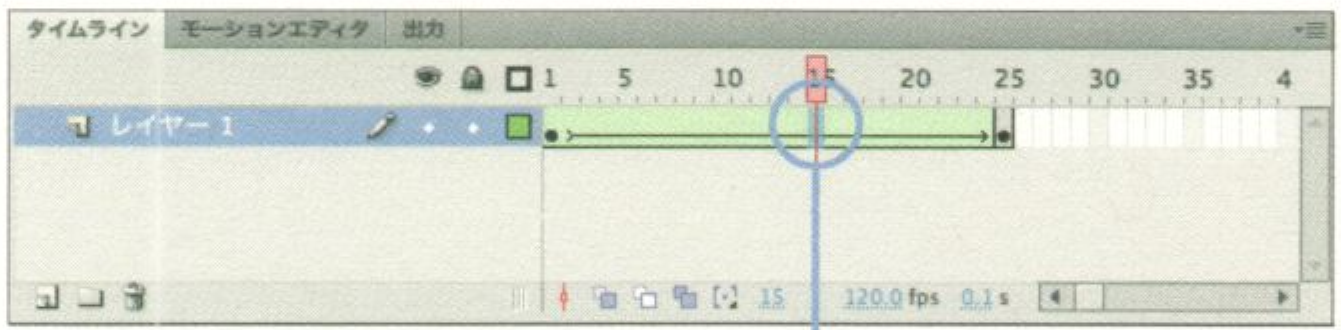
イージングの使い方

イーズイン、イーズアウト、ともに［プロパティ］パネルから設定を行います。
ここでは、シェイプトゥイーンにイージングを付けていく方法を紹介します。

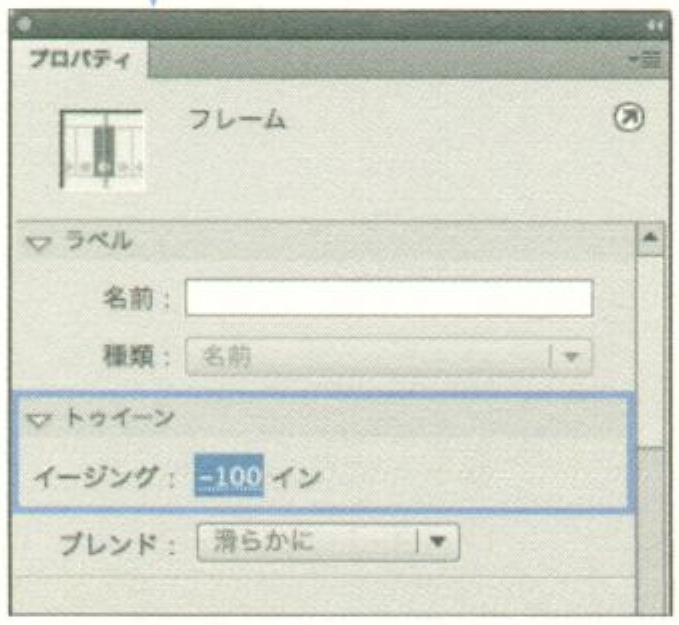
イーズイン（加速）

トゥイーンを設定したフレームを選択した状態で、［プロパティ］パネルの「トゥイーン：イージング」を「-100」にすると、次第に加速していくアニメーションになります。

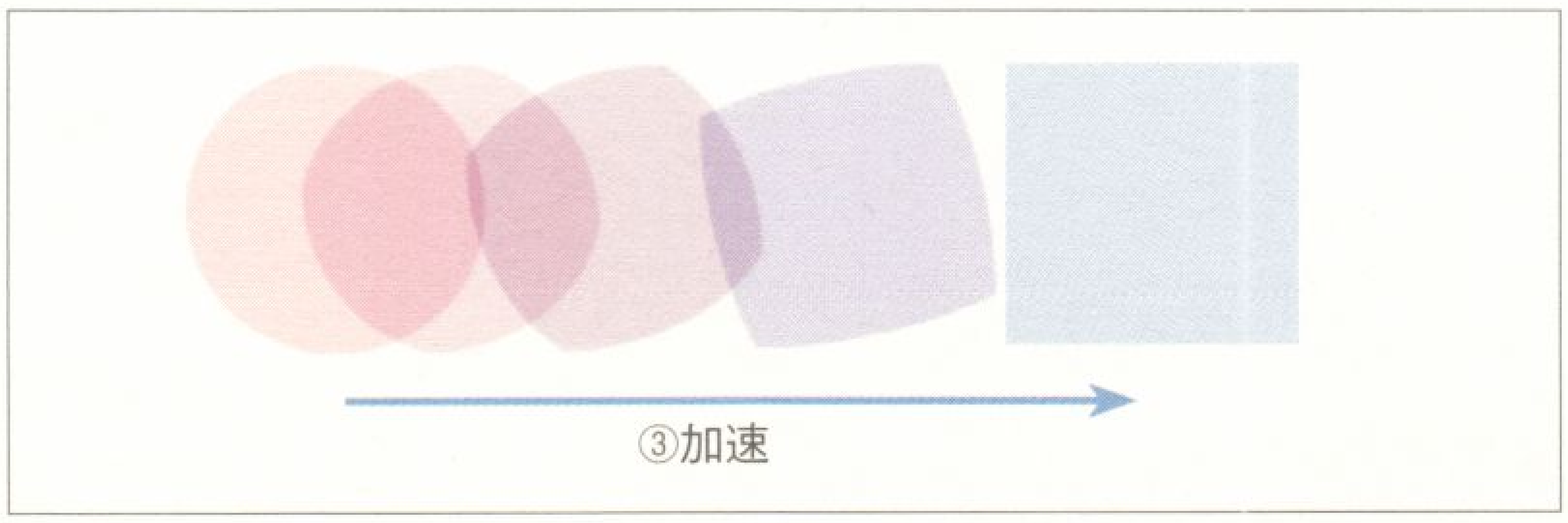
■ トゥイーン ……………………………… P047へ



①フレームを選択



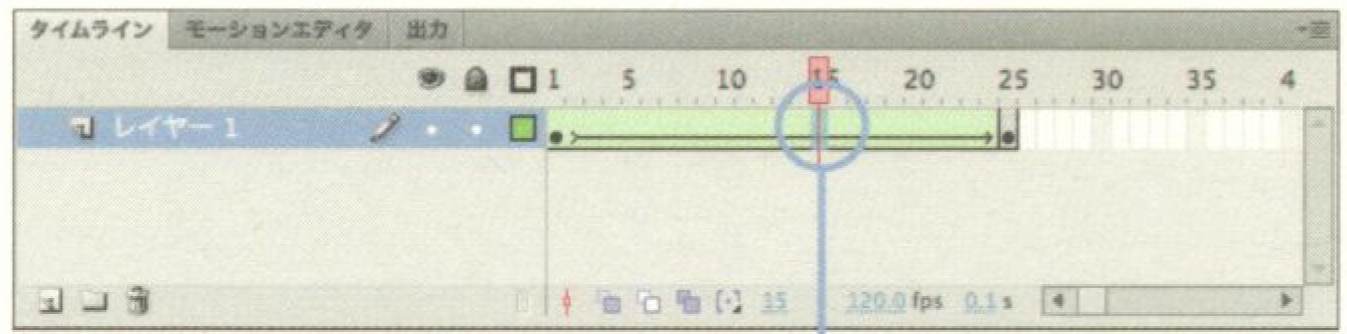
②イージングを「-100」に設定



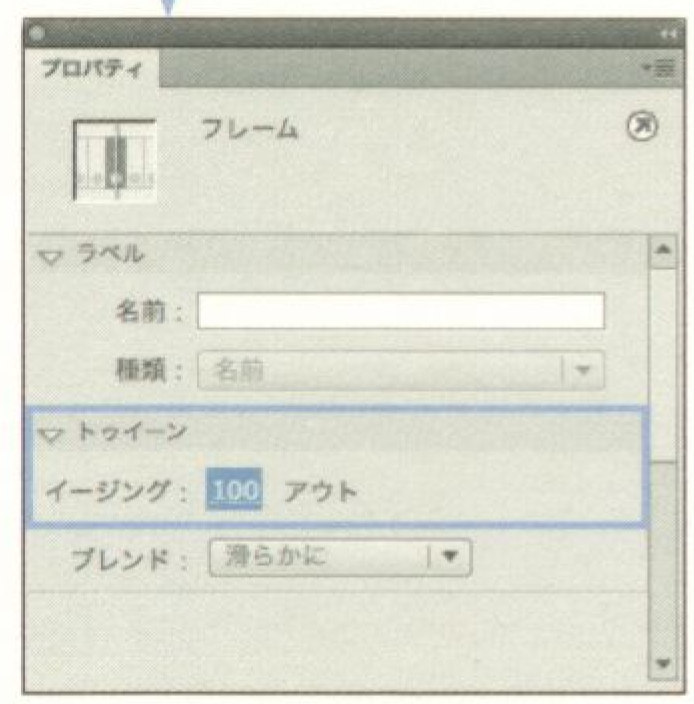
イーズアウト（減速）

トゥイーンを設定したフレームを選択した状態で、［プロパティ］パネルの「トゥイーン：イージング」を「100」にすると、次第に減速していくアニメーションになります。

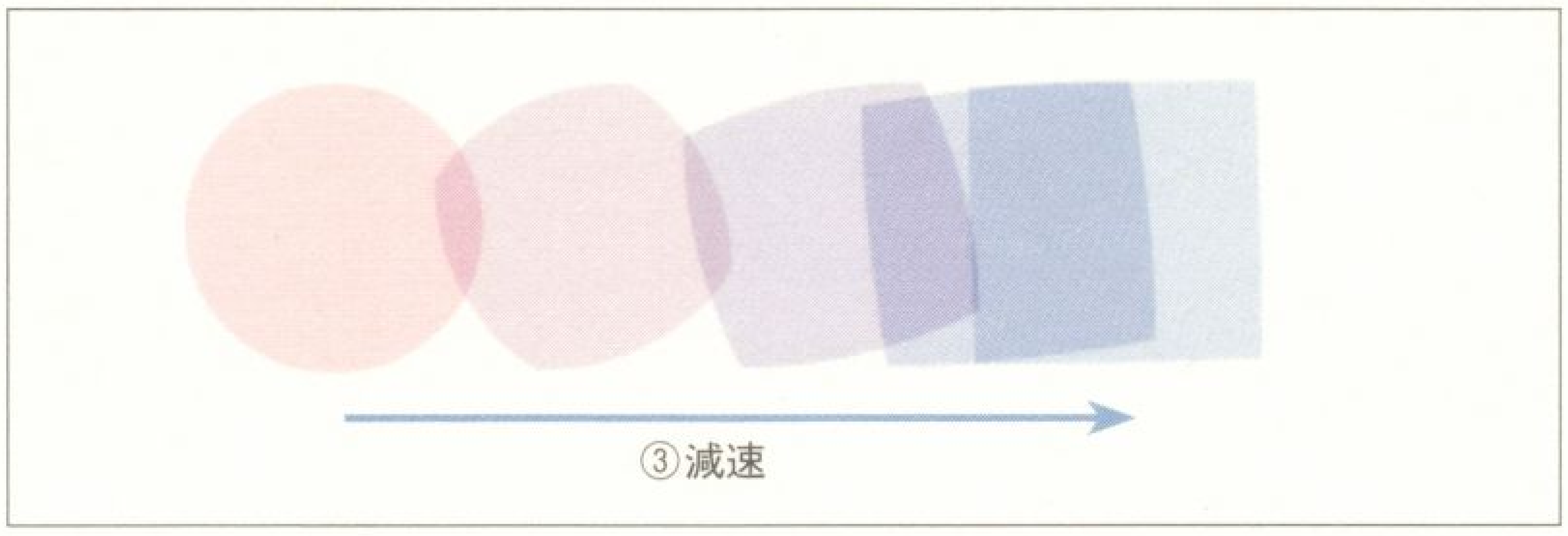
■ トゥイーン ………………………………… P047へ



①フレームを選択



②イージングを「100」に設定



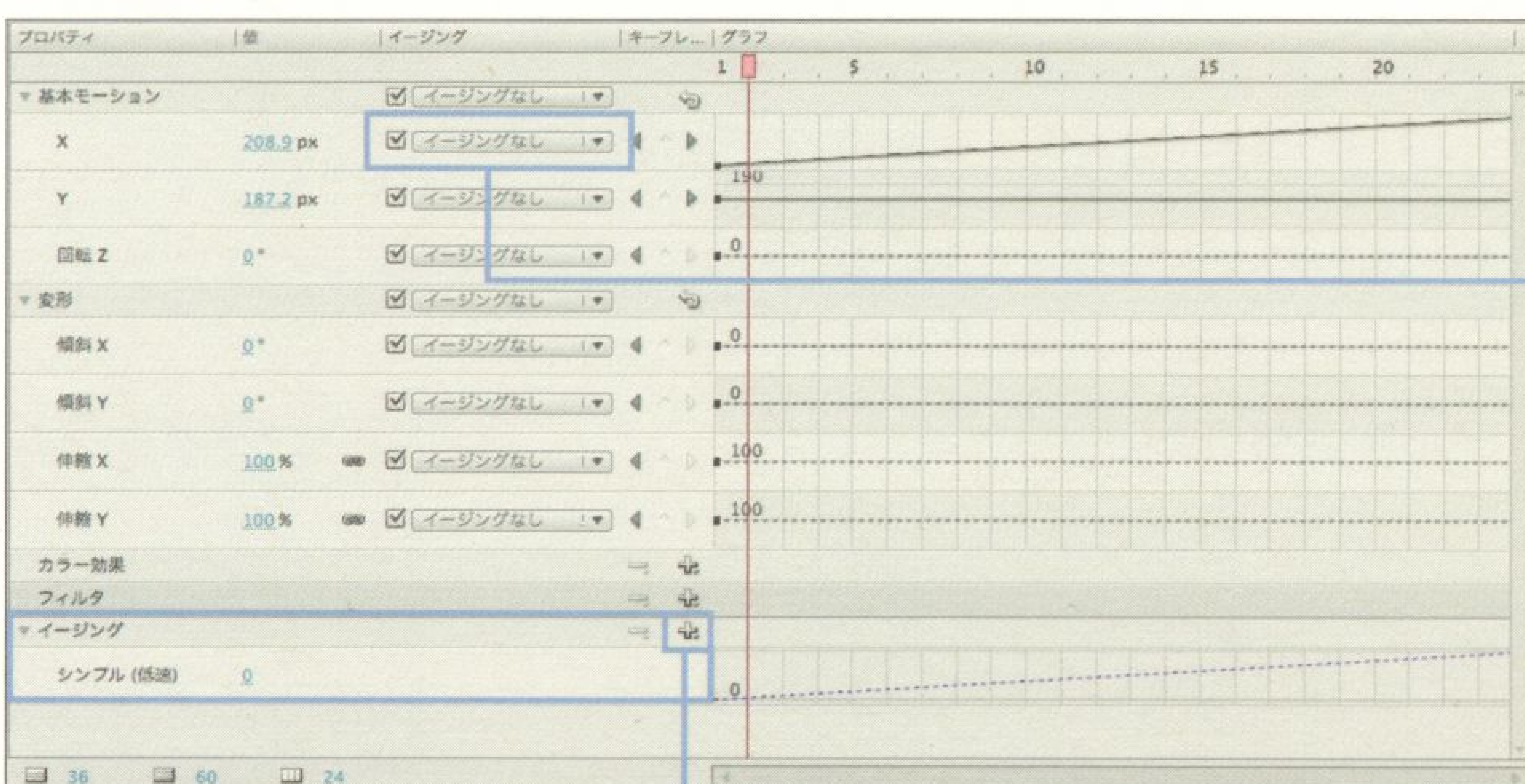
イージングの詳細設定

モーショントゥイーンとクラシックトゥイーンでは、イージングの詳細設定を行えます。

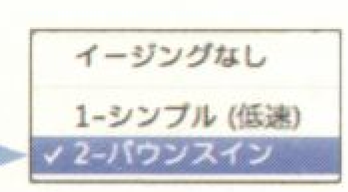
▶▶ モーショントゥイーン ── モーションエディタの利用

モーショントゥイーンにイージングを適用する際、Flash CS4から追加されたモーションエディタで、あらかじめ用意されたイージングパターンを簡単に適用することができます。
モーショントゥイーンを指定したいオブジェクトを選択ツールで選択した状態で［ウィンドウ→モーションエディタ］でモーションエディタを表示します。イージングの種類を選択し、各プロパティで個別に適用します。

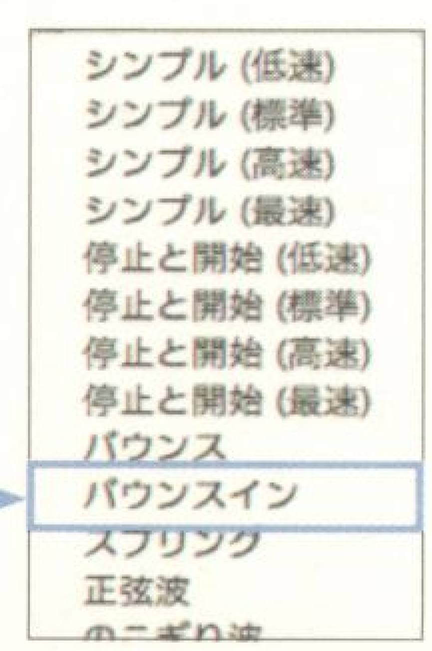
モーションエディタ

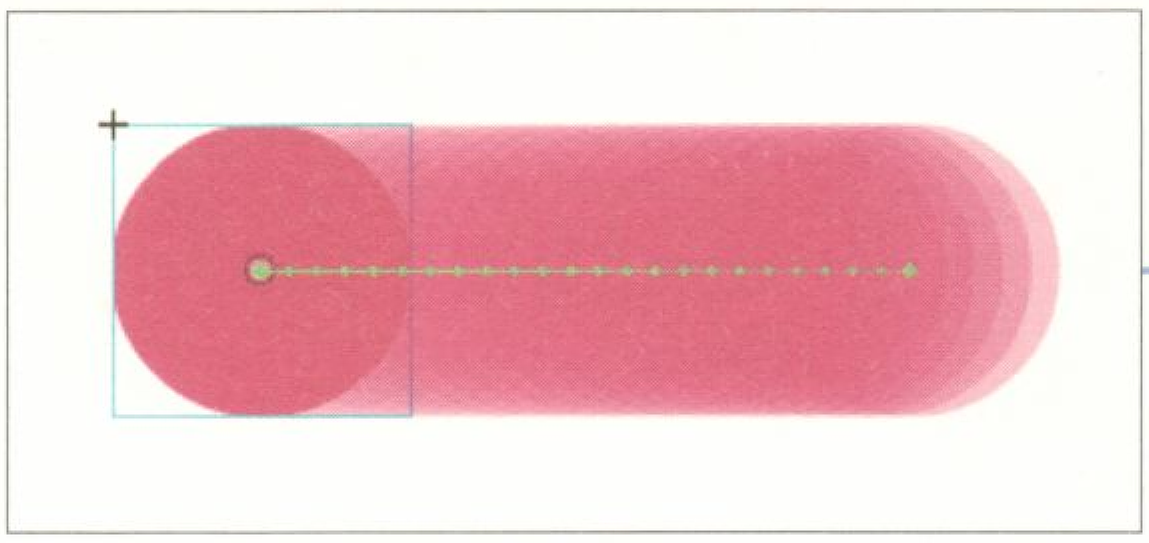


①イージングの種類を選択

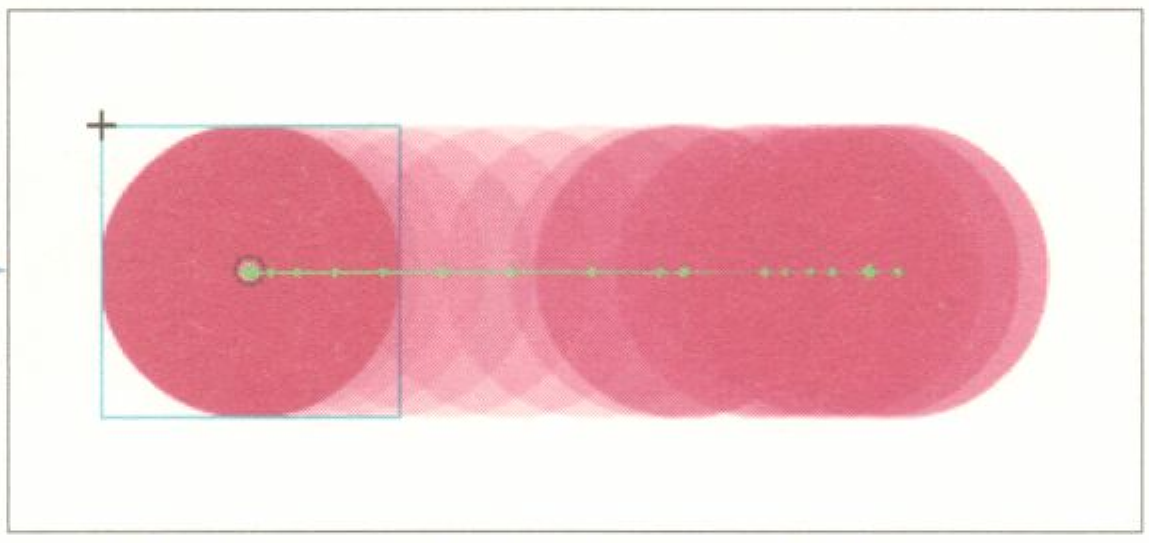


②イージングを適用





ステージ上のモーショントゥイーン



③トゥイーンアニメにイージングが適用される

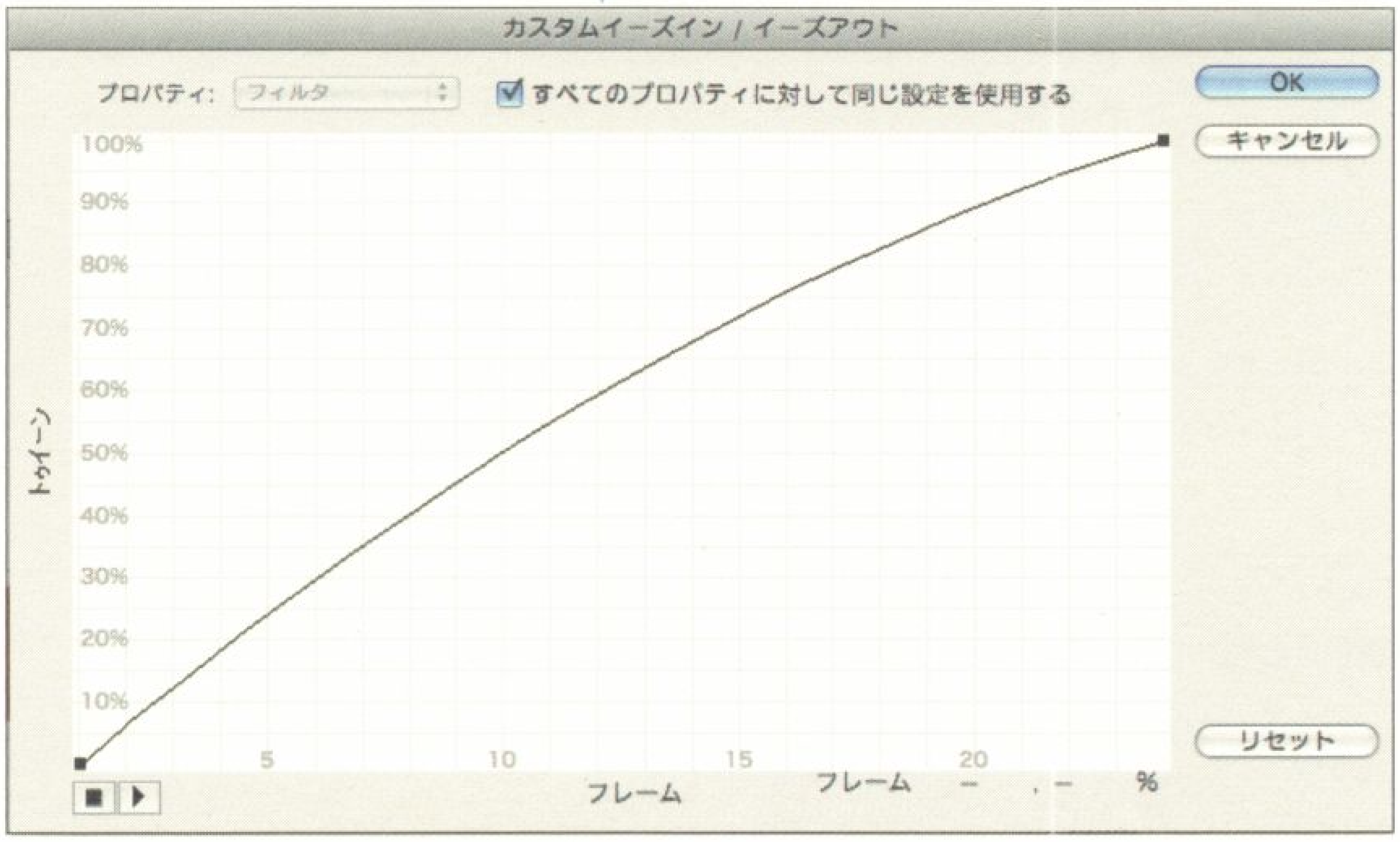
▶▶ クラシックトゥイーン ── カスタムイーズイン/イーズアウト

クラシックトゥイーンでも、イージングの詳細設定を行うことが可能です。[プロパティ] パネルの「トゥイーン」にある [イージングを編集] のアイコンをクリックします。「カスタムイーズイン/イーズアウト」ウィンドウが表示されます。

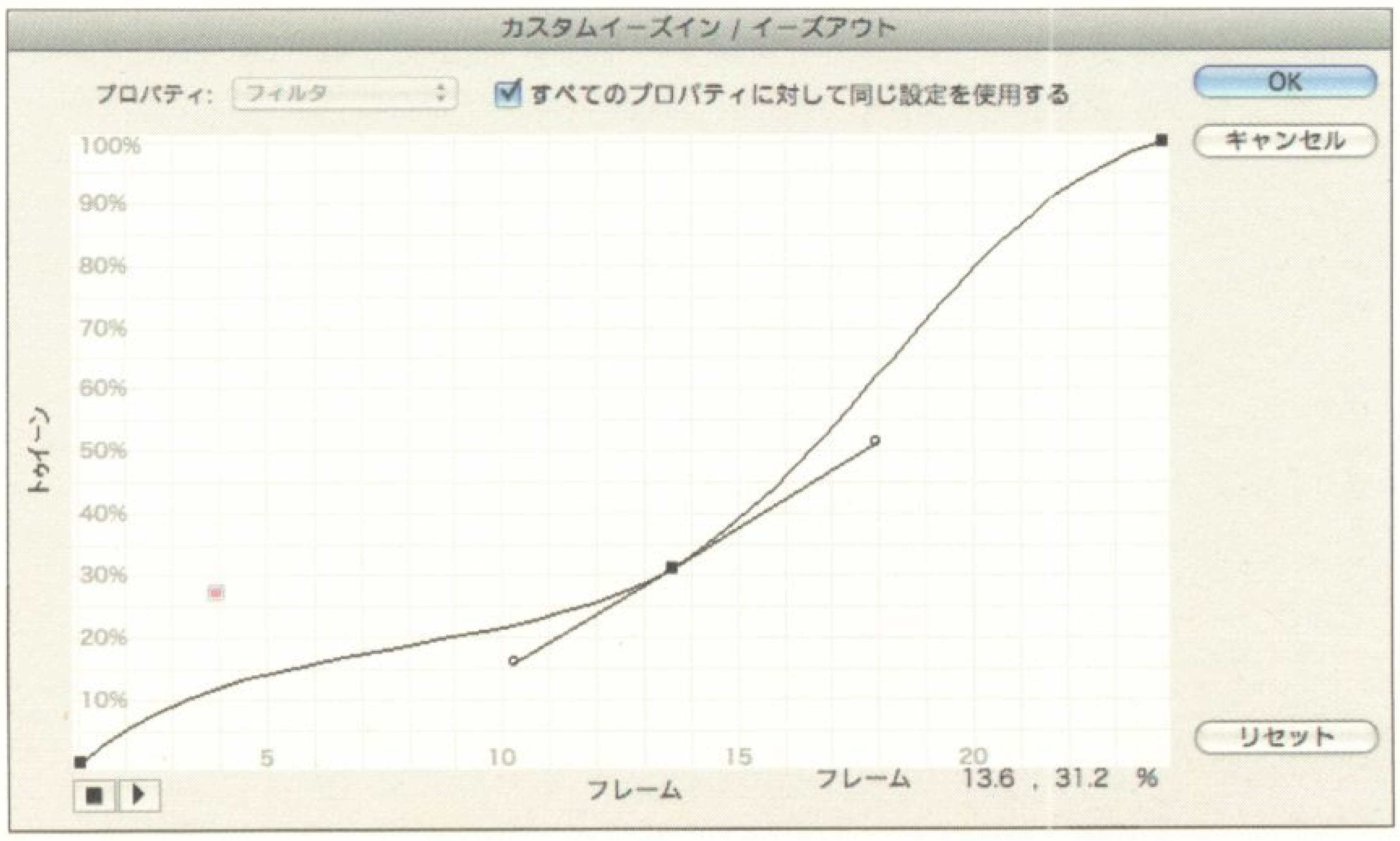
プロパティパネル



①「イージングを編集」をクリック



②カスタムイーズイン/イーズアウトのウィンドウが表示される



③曲線をドラッグしてイージングを設定する

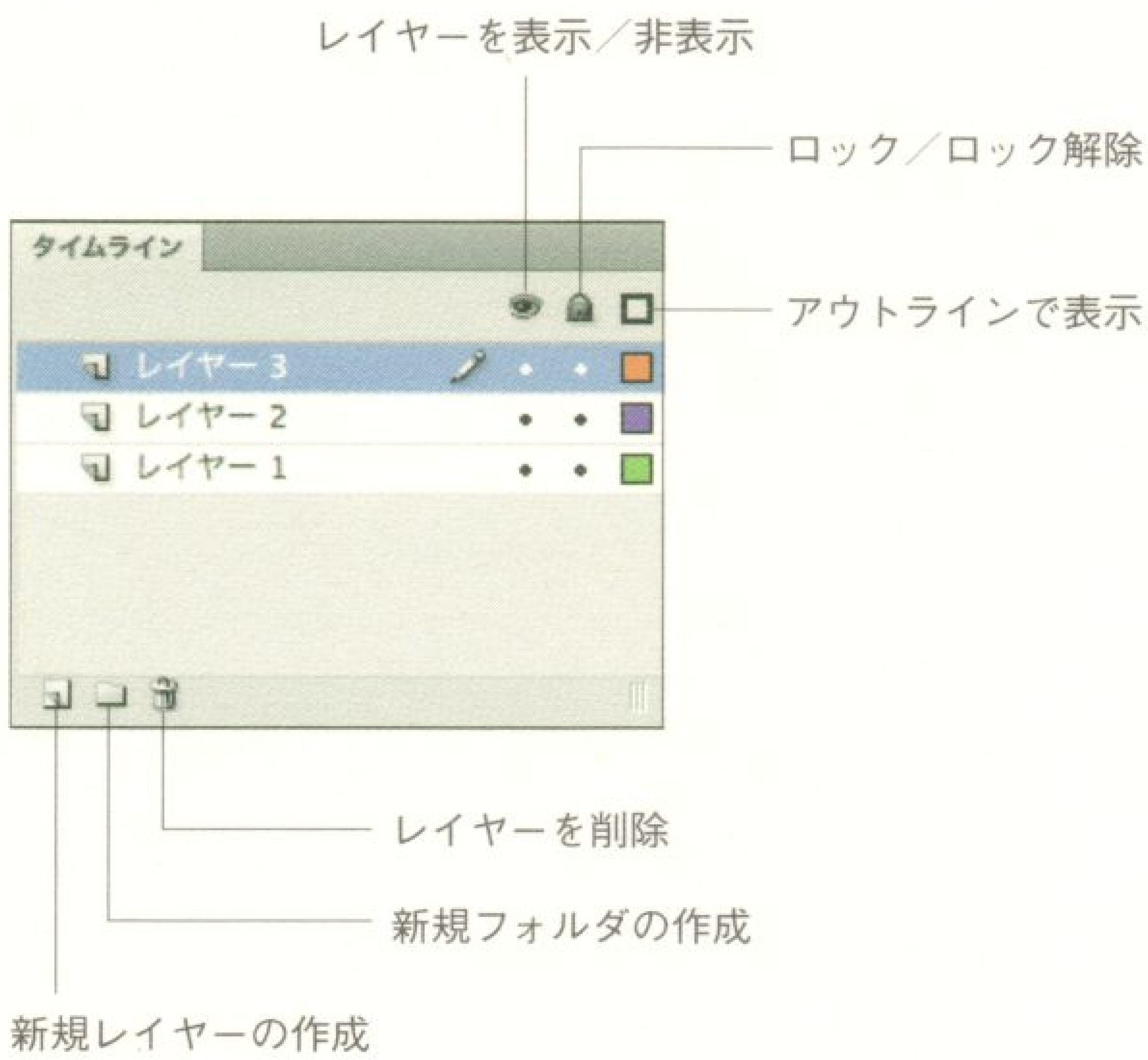
LECTURE 1-8

Flashの基本

レイヤー

レイヤーは必要に応じて、削除・追加することができます。タイムラインでレイヤーを追加したりフォルダにまとめられる他、レイヤーのロック/ロック解除や、表示を行います。

[タイムライン] パネルにおけるレイヤーの構成は、このようになっています。

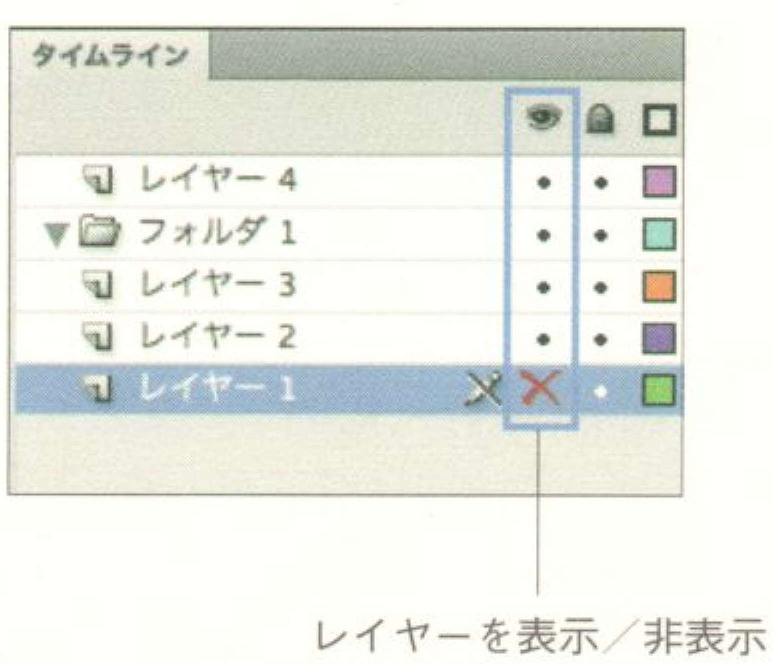


レイヤーの機能

レイヤーがたくさん作成されてくると、管理が大変になります。
そういった際に便利な、レイヤー機能を紹介します。

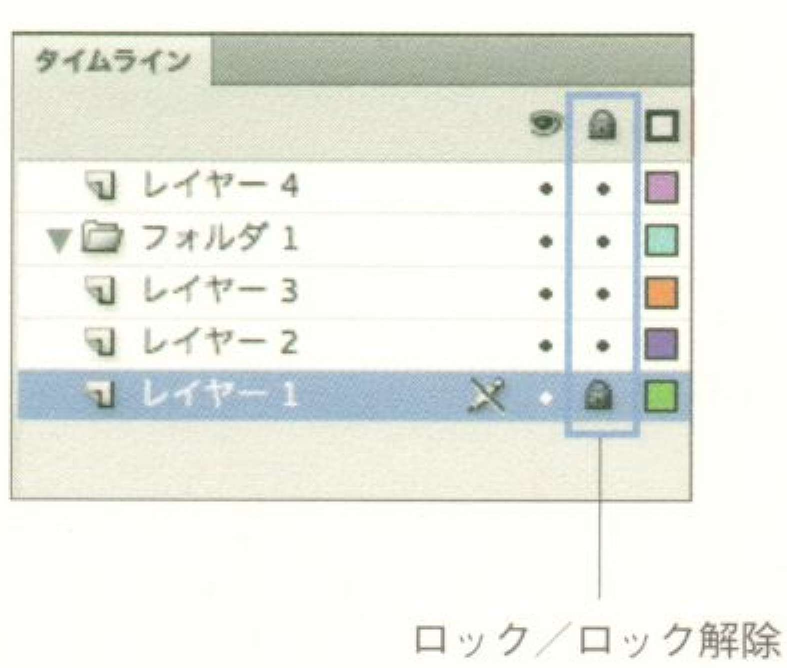
レイヤーの表示／非表示

レイヤーの[目のアイコン]欄をクリックで、×印が表示され、レイヤー全体の表示／非表示を切り替えられます。



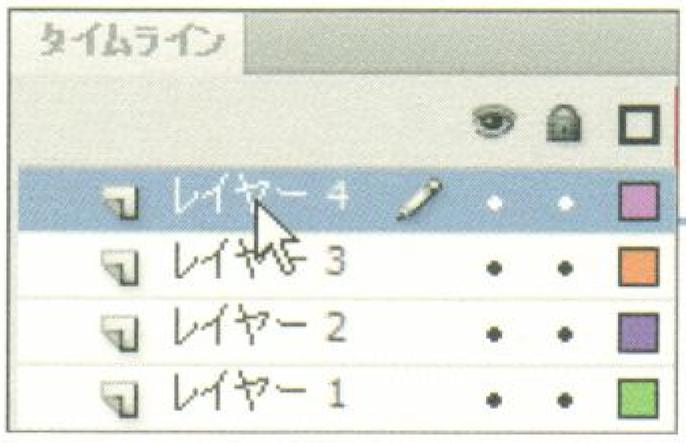
ロック／ロック解除

レイヤーの[鍵のアイコン]欄をクリックで、鍵印が表示され、レイヤー全体のロック／ロック解除ができます。

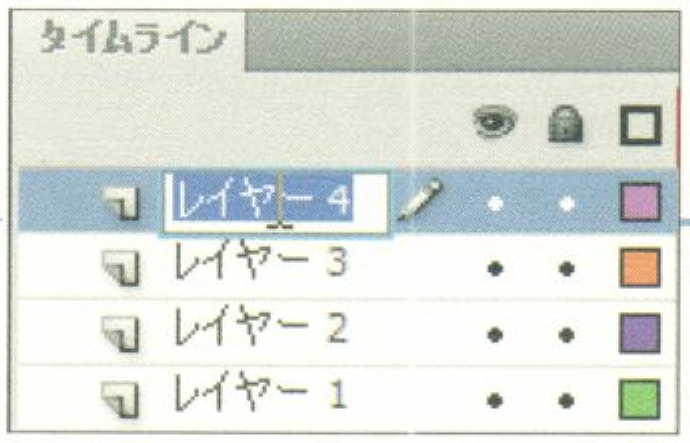


レイヤー名の変更

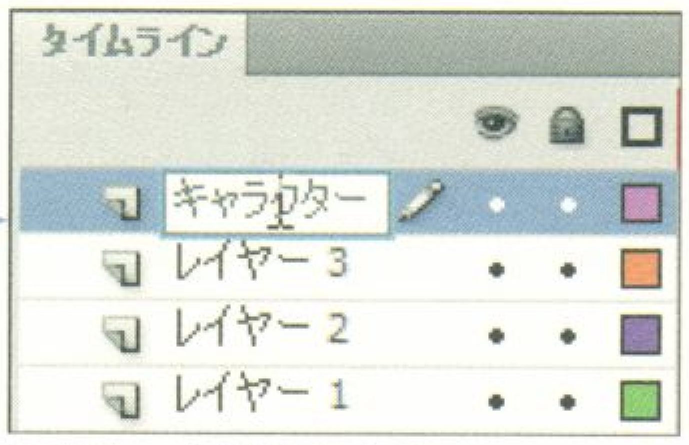
レイヤーの名前を変更するには、レイヤー名のところをダブルクリックします。レイヤーの名前が編集できる状態になりますので、新しい名前を入力します。



①レイヤー名の上にマウスカーソルを持って行く



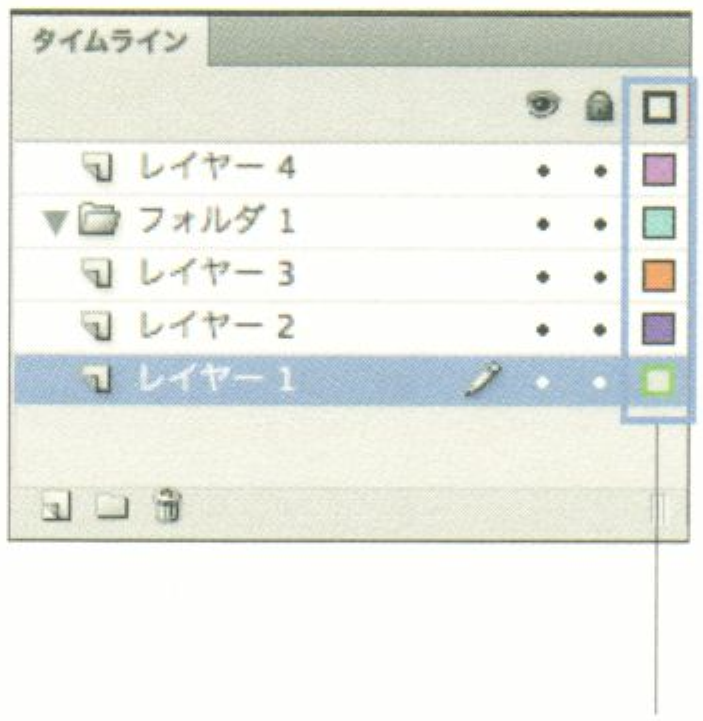
②ダブルクリックで名前が編集可能になる



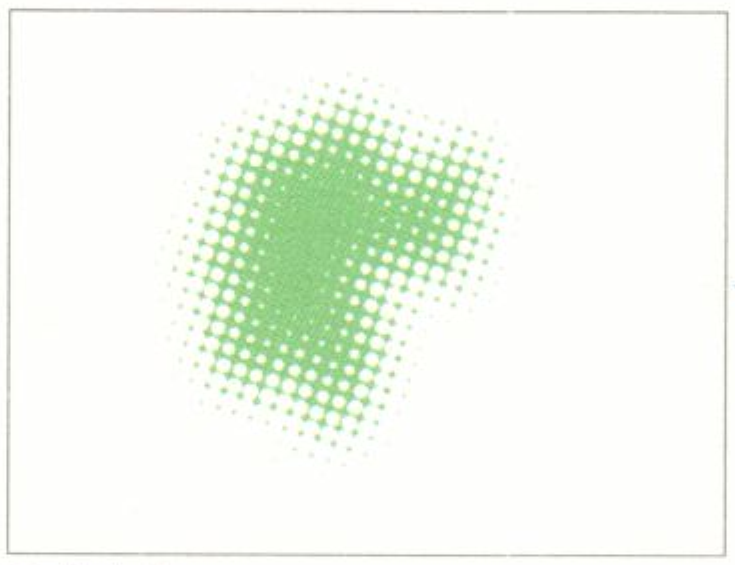
③新しい名前を入力する

レイヤーをアウトラインで表示

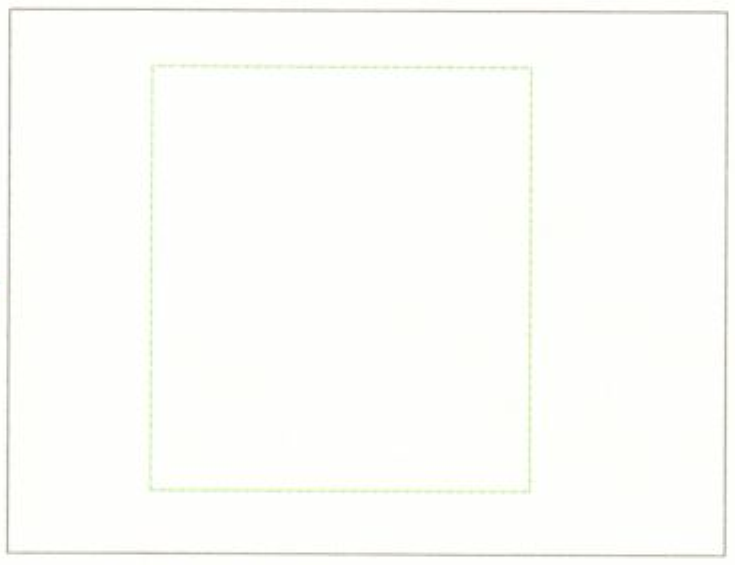
レイヤーの□欄をクリックで、■が枠だけの□表示になります。オブジェクトのアウトラインだけが表示されるようになります。



レイヤーをアウトラインで表示



通常表示



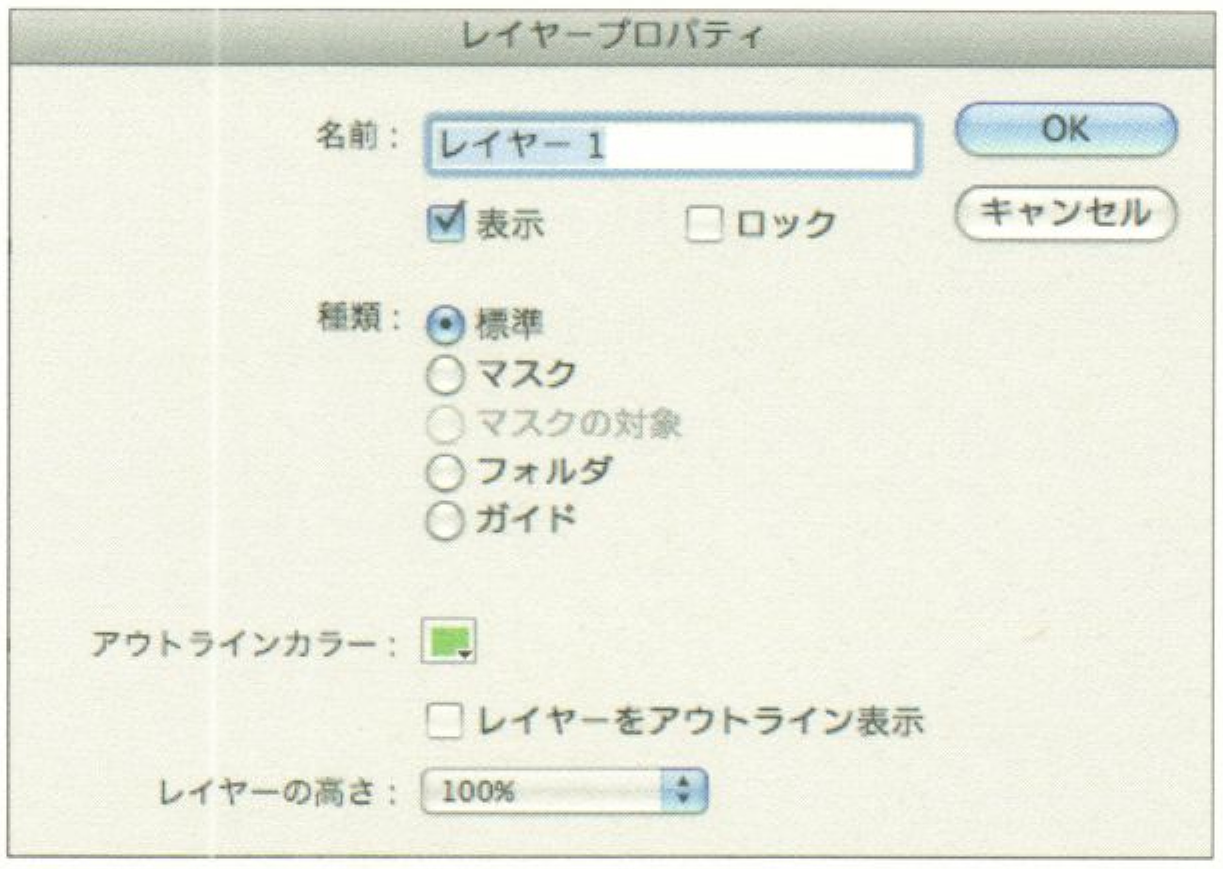
アウトライン表示

詳細設定

レイヤーの名前の前にあるアイコン( )をダブルクリックすると、レイヤープロパティのウィンドウが表示され、詳細設定を行うことができます。



①ダブルクリック



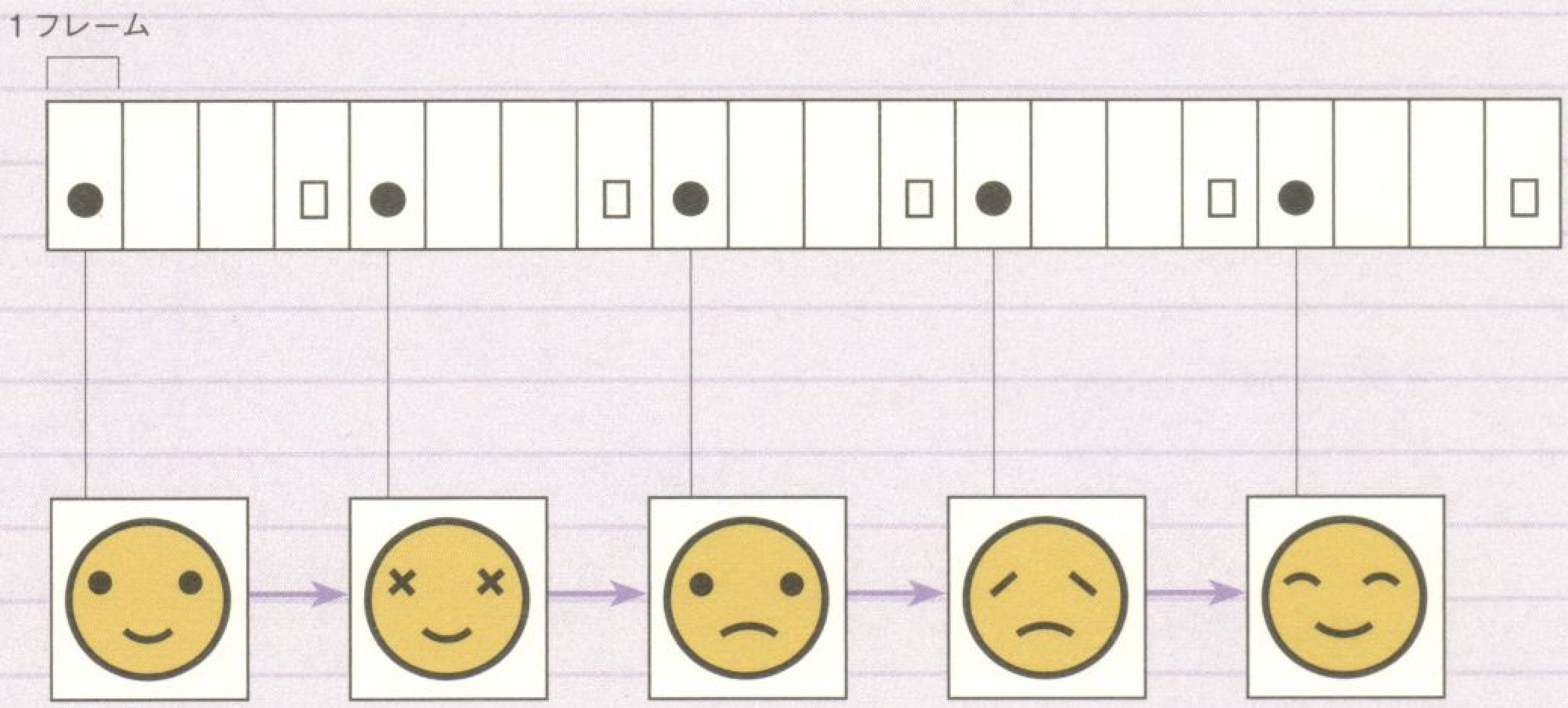
②レイヤープロパティが表示される

LECTURE 2-1

Flashによる表現

アニメーションを作る

タイムラインに表示させるフレームは、アニメーションの1コマになります。この、タイムライン上で連続して再生されるフレームによってアニメーションが構成されています。アニメーションには、「フレーム・アニメーション(パラパラ漫画のように1フレームごとに異なった絵の連続でできたアニメ)」と「トゥイーンアニメーション(最初と最後の絵の間の動きを自動的に補完してくれるアニメ)」があります。

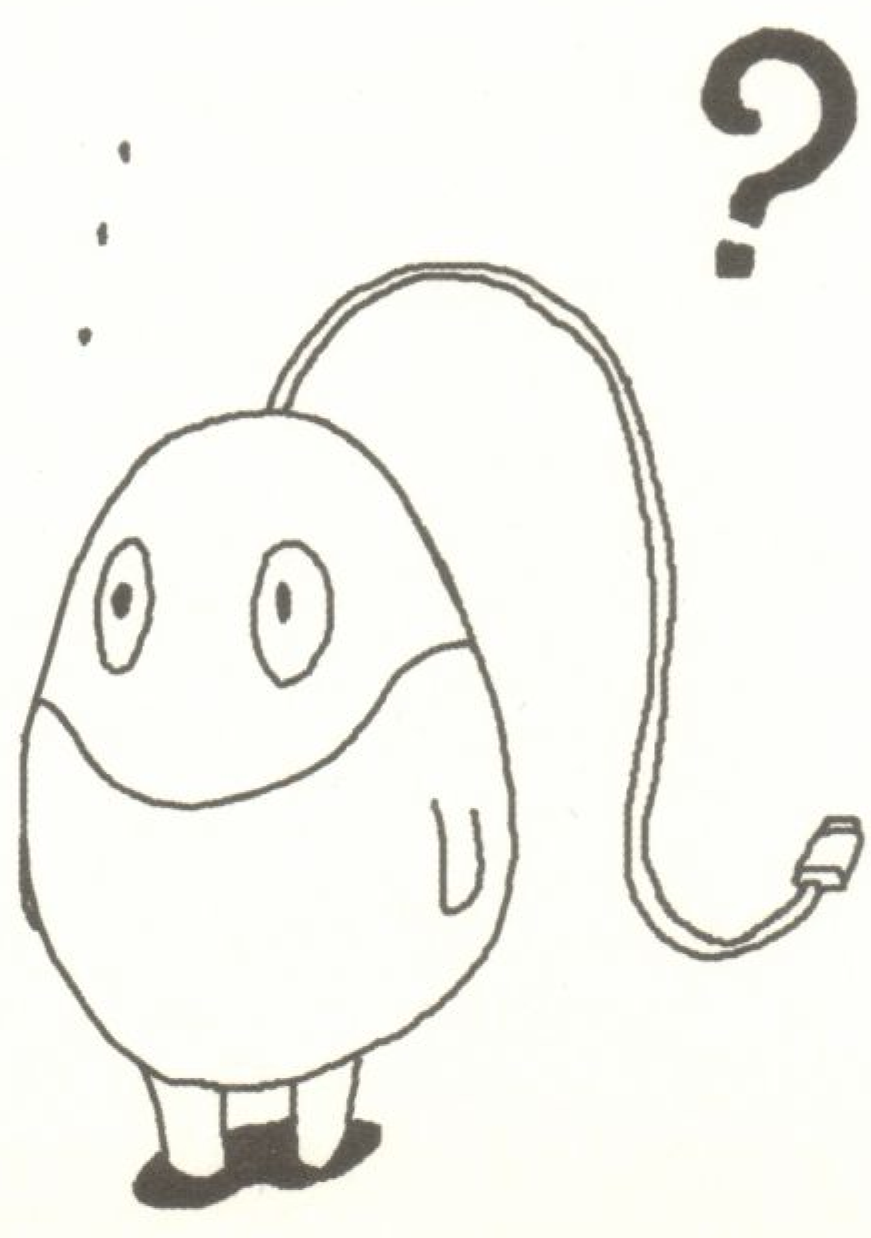


アニメーションが使われているサンプル

■ PRACTICE 1：
コマ全体のアニメーションを作る … P180へ

合わせて読むと便利な項目

■ シンボル ……………………… P034へ
■ イージング ……………………… P041へ

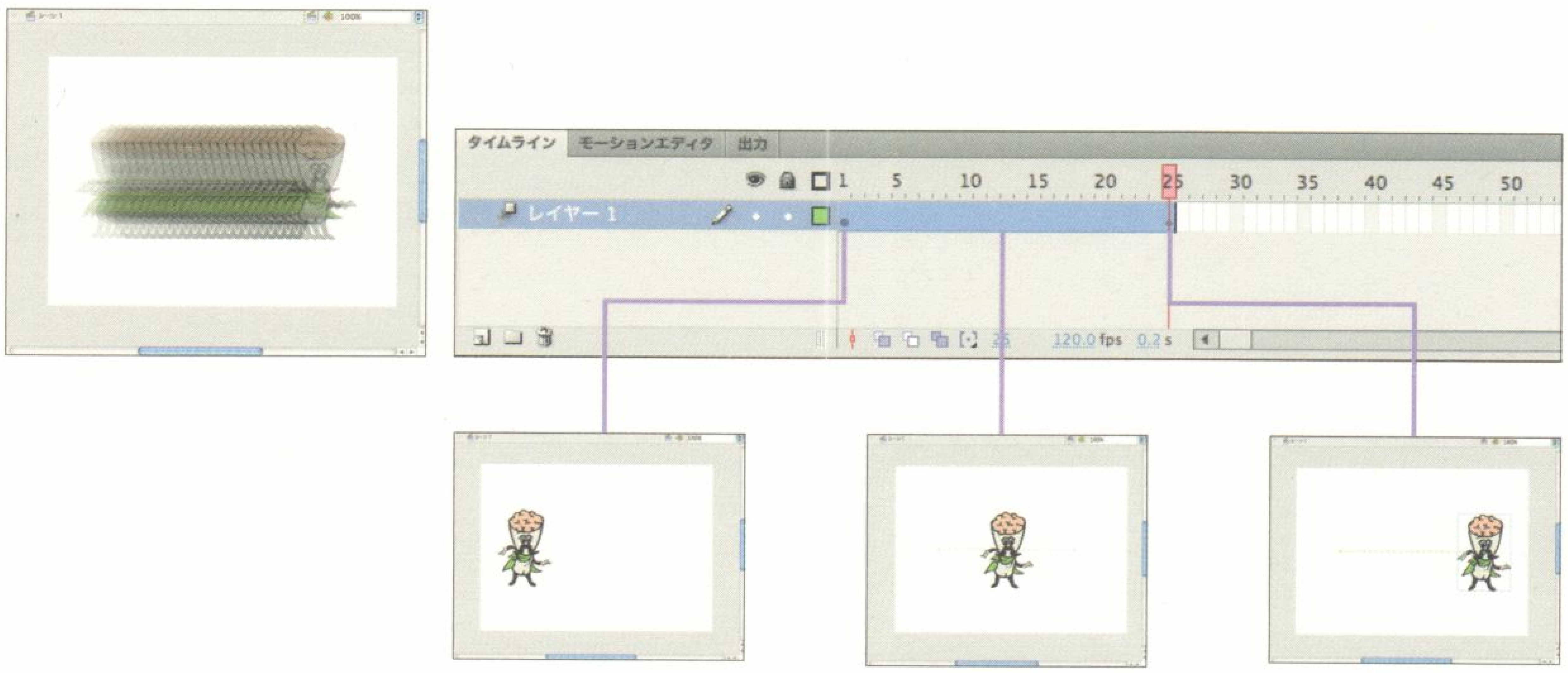


モーショントゥイーンとシェイプトゥイーン

それでは具体的にトゥイーンアニメーションについて説明していきましょう。
トゥイーンには「モーショントゥイーン」と「シェイプトゥイーン」の2種類があります。

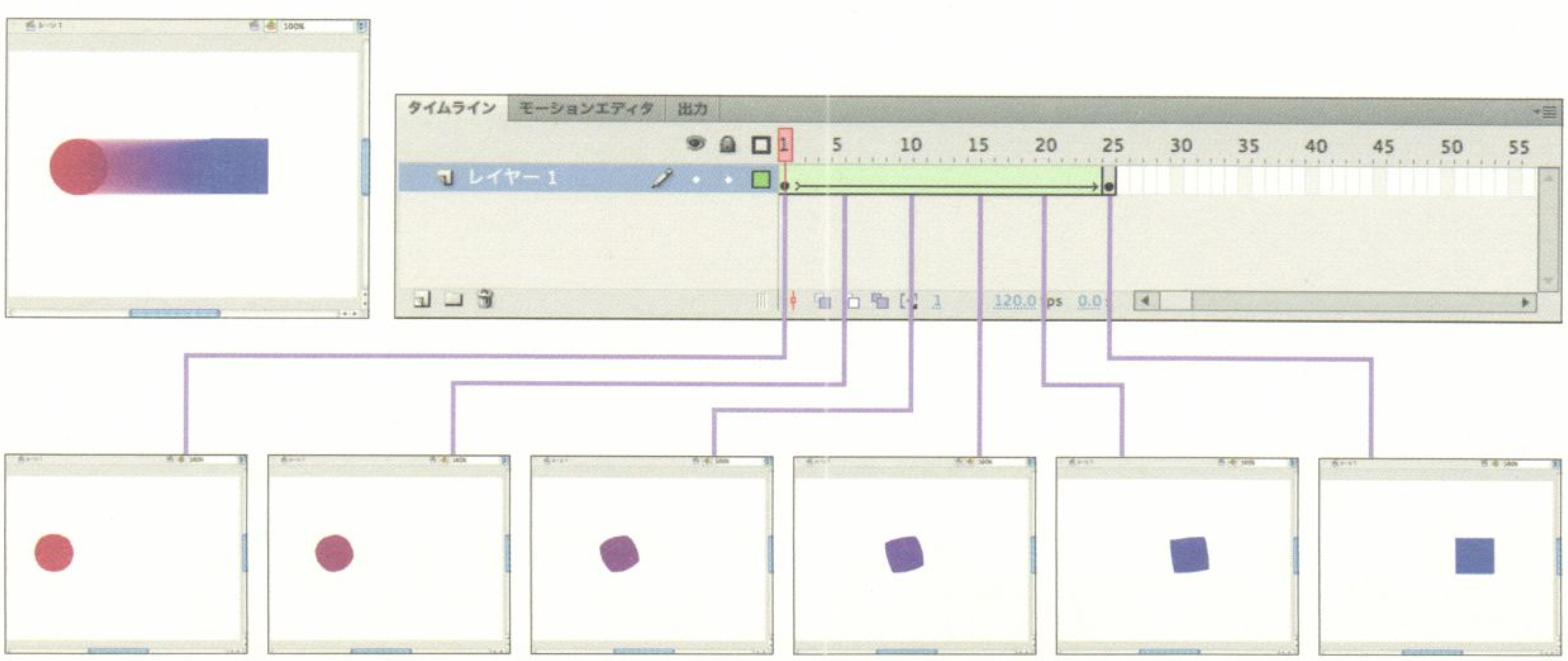
▶▶ モーショントゥイーン

モーショントゥイーンは、グループ化またはシンボルに変換されたオブジェクトに対して設定可能なアニメーションです。モーショントゥイーンは、オブジェクトの大きさや角度、傾斜や位置などのプロパティを自動的に変化させるときに利用します。Flash CS3までのモーショントゥイーンは「クラシックトゥイーン」と名前が変更されています。モーショントゥイーンとクラシックトゥイーンの機能面の違いについては、次ページで解説します。



▶▶ シェイプトゥイーン

シェイプトゥイーンとは、2つのキーフレーム間で設定します。
それぞれのキーフレームに配置したシェイプの位置や形状、カラーをトゥイーンします。



▶▶ モーショントゥイーン、クラシックトゥイーンの違い

クラシックトゥイーンはFlash CS3以前のFlashで作成されたトゥイーンのことを指し、キーフレームごとにインスタンス、グループおよびタイプの位置、サイズ、回転、傾斜をトゥイーンできます。CS4のモーショントゥイーンではトゥイーンを細かく制御できますが、クラシックトゥイーンではフレームスクリプトを使用できたり、グラフィックシンボルのフレーム数の設定ができるなどの、特定の機能がある点で優位です。

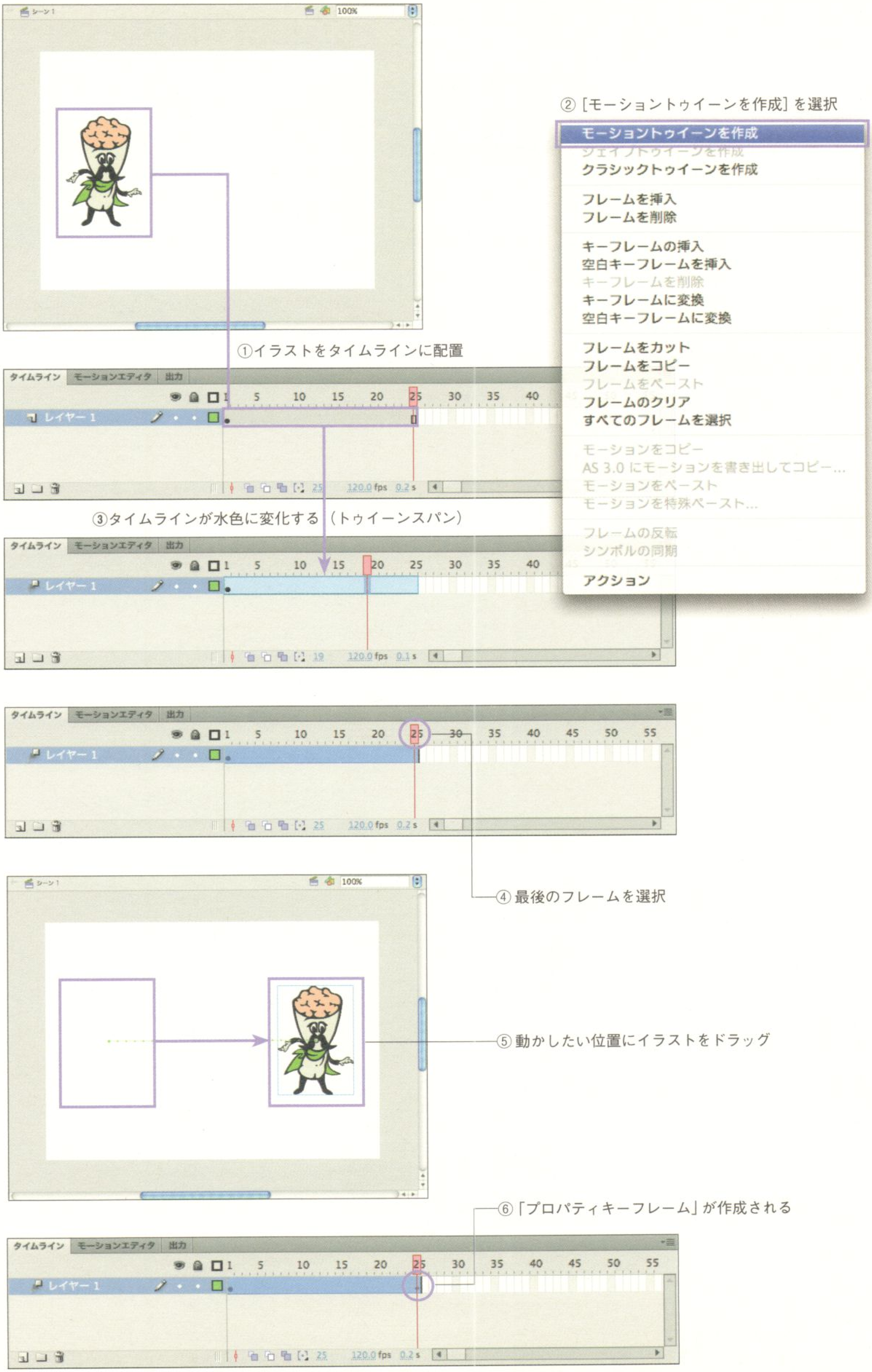
項目	モーショントゥイーン	クラッシックトゥイーン
キーフレーム	使用しない	使用する
トゥイーン作成に必要なオブジェクト数	1つのオブジェクトで作成することができる	2つのオブジェクト間で作成することができる
テキストのアニメーション	テキストオブジェクトそのままでトゥイーンを作成することが可能	グラフィックシンボルに変換される
フレームスクリプト	使用できない	使用できる
タイムライン上での伸縮とサイズ変更	単一オブジェクトとして扱われるので変更が可能	フレーム単位で個別の変更が可能
イージングの適用	モーショントゥイーンスパン全体に適用される（特定のフレームのみイージングするには、カスタムイージング曲線の作成が必要）	キーフレーム間のフレームに適用される
カラー効果の適用	各トゥイーンにつき カラー効果を1つだけ適用できる	着色やアルファ透明度などのカラー効果を2つの異なった設定間で適用できる
「X」、「Y」、および「Z」軸を持つ3D オブジェクトのアニメーション化	できる	できない
グラフィックシンボルのフレーム数の増減	設定できない	設定できる

トゥイーンの作成方法

モーショントゥイーンとクラシックトゥイーンそれぞれの作成方法を解説します。

▶▶ モーショントゥイーンの作成（次ページの図参照）

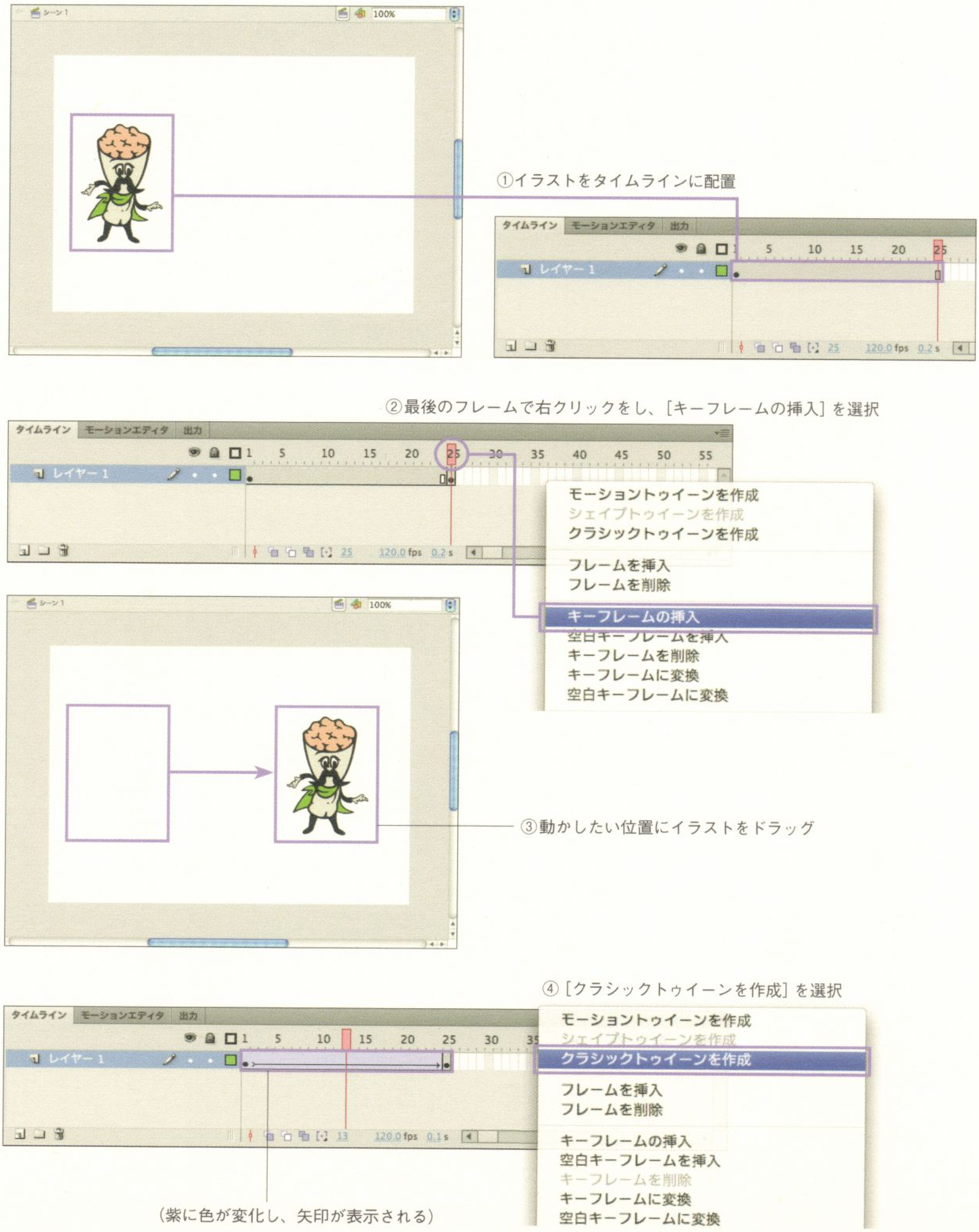
①イラストをタイムラインに配置します。
②右クリックで［モーショントゥイーンを作成］を選択します。
③タイムラインが水色に変化します。この状態を「トゥイーンスパン」と言います。
④作成された「トゥイーンスパン」の最後のフレームを選択します。
⑤動かしたい位置にメインステージのイラストをドラッグします。
⑥最後のフレームに印が付けられます。これをプロパティキーフレームと言います。


②［モーションツイーンを作成］を選択
モーショントゥイーンを作成
クラシックトゥイーンを作成
フレームを挿入
フレームを削除
キーフレームの挿入
空白キーフレームを挿入
キーフレームに変換
空白キーフレームに変換
フレームをカット
フレームをコピー
フレームのクリア
すべてのフレームを選択
アクション
①イラストをタイムラインに配置
③タイムラインが水色に変化する（トゥイーンスパン）
④最後のフレームを選択
⑤動かしたい位置にイラストをドラッグ
⑥「プロパティキーフレーム」が作成される
タイムライン
モーションエディタ
出力
レイヤー 1

LECTURE 2-1

▶▶ クラシックトゥイーンの作成

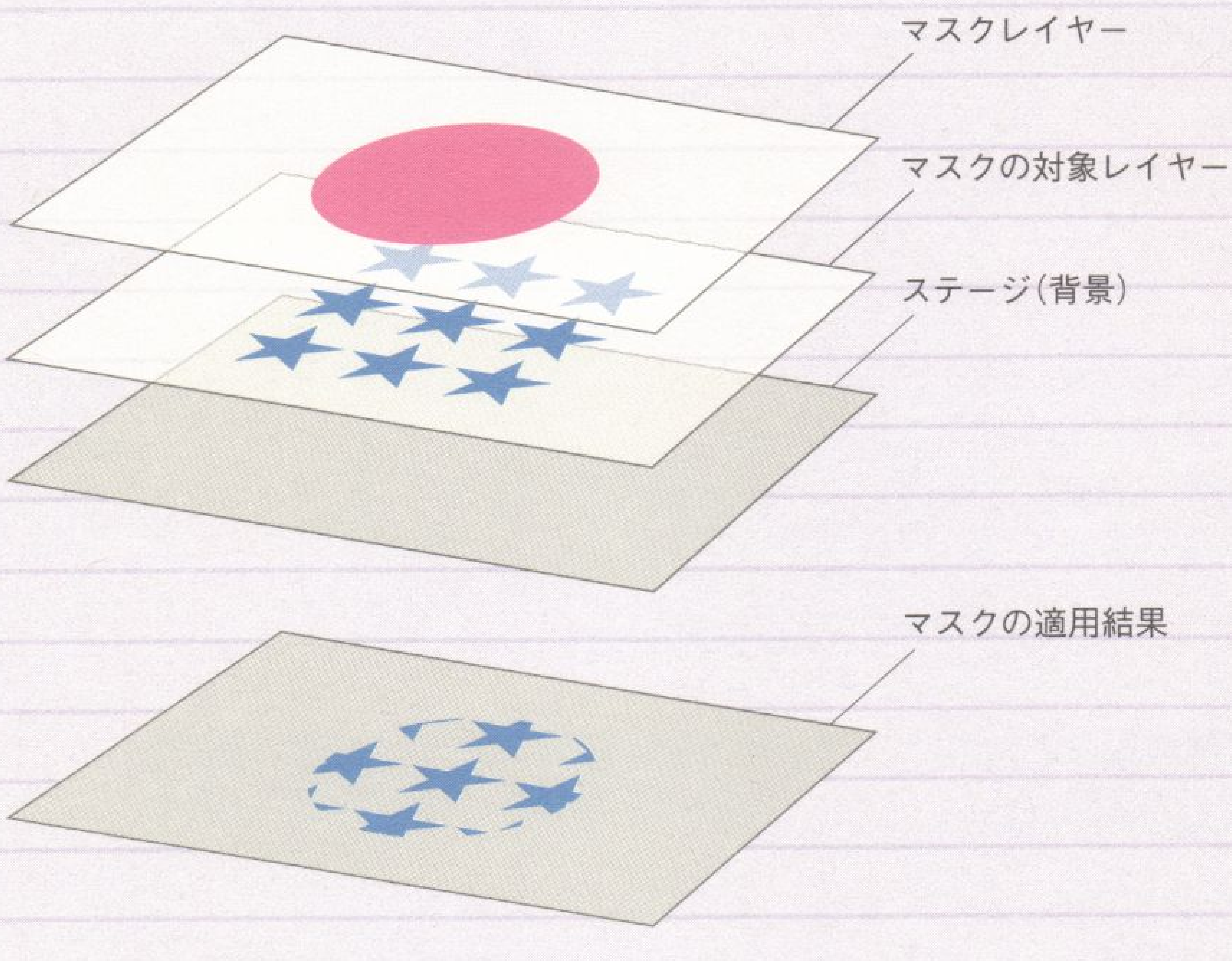
①イラストをタイムラインに配置します。
②動きの最後のフレームで右クリックをし、[キーフレームの挿入] を選択します。
③ステージ上のイラストを動かしたい最後の位置に移動します。
④キーフレームの間のフレームのいずれかを選択した状態で右クリックをし、[クラシックトゥイーンを作成] を選択します。



LECTURE 2-2

Flashによる表現 マスクを使う

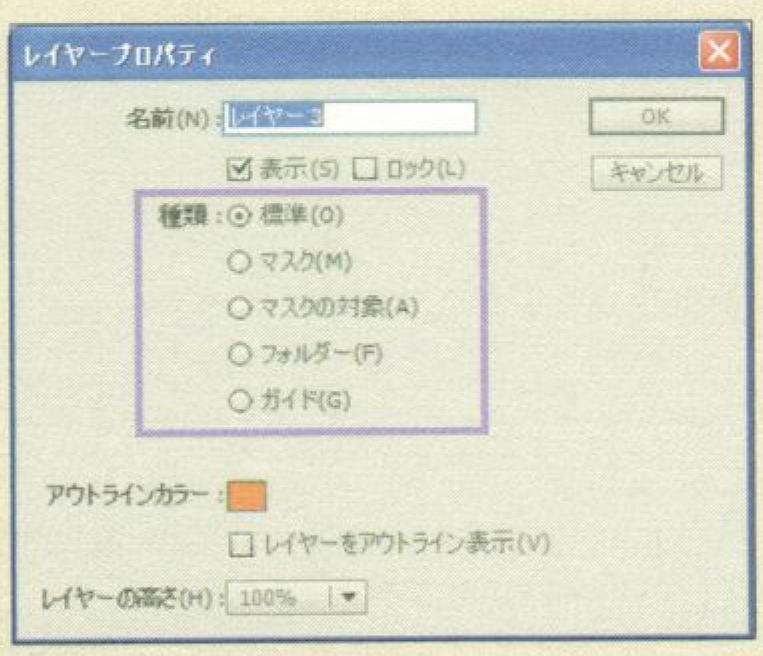
オブジェクトを部分的に表示したい場合、その範囲を指定する機能が「マスク」です。「マスク」は、表示したいグラフィックと、その対象となるオブジェクトを別々のレイヤーに配置することで、作成できます。



マスクが使われているサンプル

■ PRACTICE 1：マスクを作成する ……………………………………………………P186へ

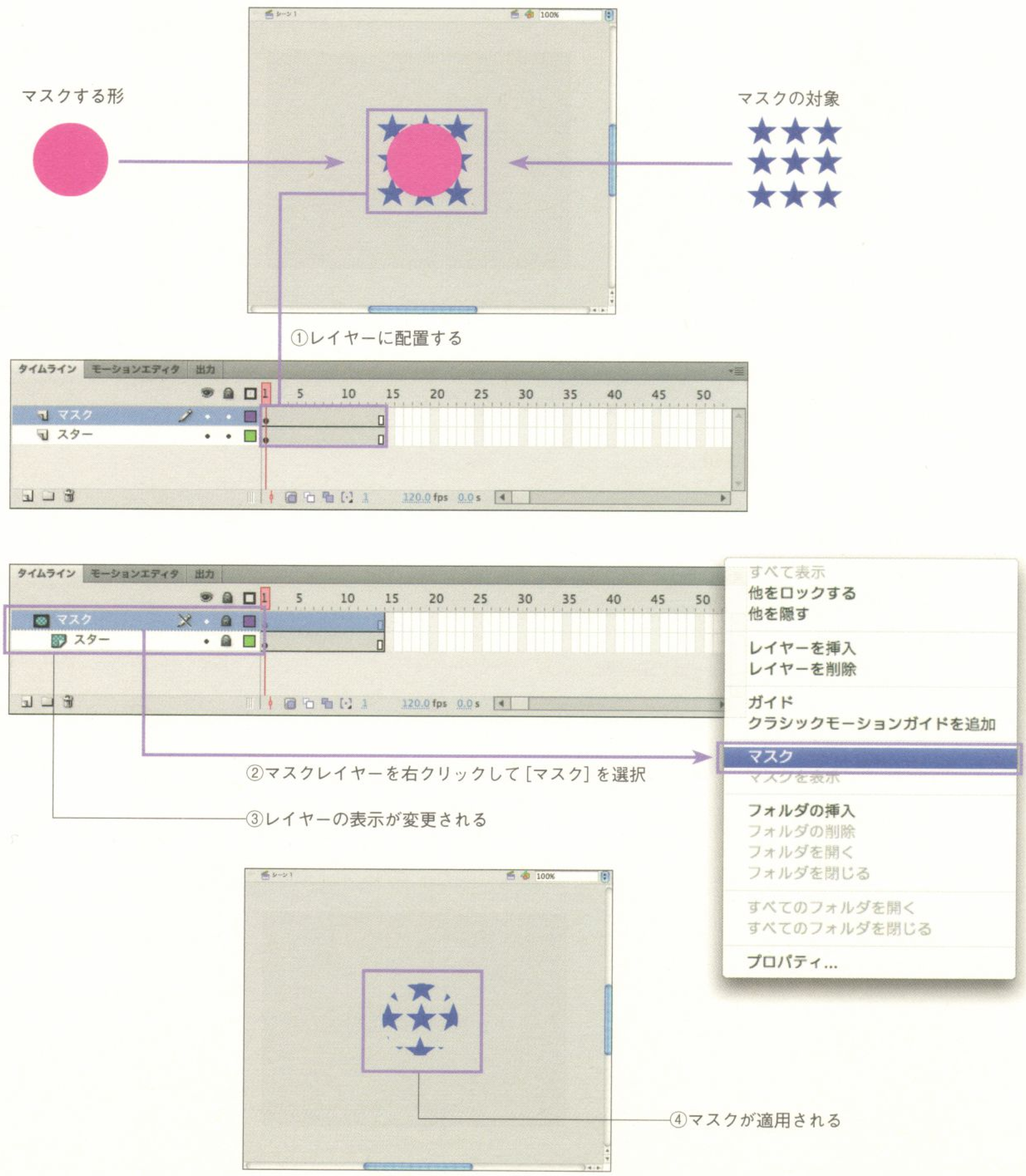
レイヤーをマスクレイヤーにしたり外したりする操作は、次ページの方法以外に［レイヤープロパティ］からも行えます。
レイヤー名の前にあるアイコンをダブルクリックして［レイヤープロパティ］を表示します。
「種類」の「マスク」を選択するとマスクレイヤーに、「マスクの対象」を選択するとマスクレイヤーに従ってマスクされるレイヤーになります。「標準」をクリックすると通常のレイヤーに戻ります。



マスクの使い方

マスクは、オブジェクトそのものは変更せずに、一部分だけを表示する際に使用します。
例えば窓枠をマスクレイヤーにして、外を流れる風景にマスクをかけて表示させる、などの例が挙げられます。

①星型のシェイプのレイヤーの上に、新規レイヤーでシェイプとなる円形を描いたレイヤーを作成します。
②マスクレイヤーを右クリックして［マスク］を選択します。
③レイヤーの表示が変更されます。
④マスクが適用され、オブジェクトを配置した部分だけ、星形シェイプが表示されます。



LECTURE 2-3

Flashによる表現

画像の扱い方

Flashでは画像ファイルをライブラリに読み込むことで、それを画面に配置したり、シンボル化することが可能です。ここでは、画像をライブラリに読み込み、使用する方法を説明します。

合わせて読むと便利な項目

- シンボル ……………………P034へ
- Loader ……………………P158へ

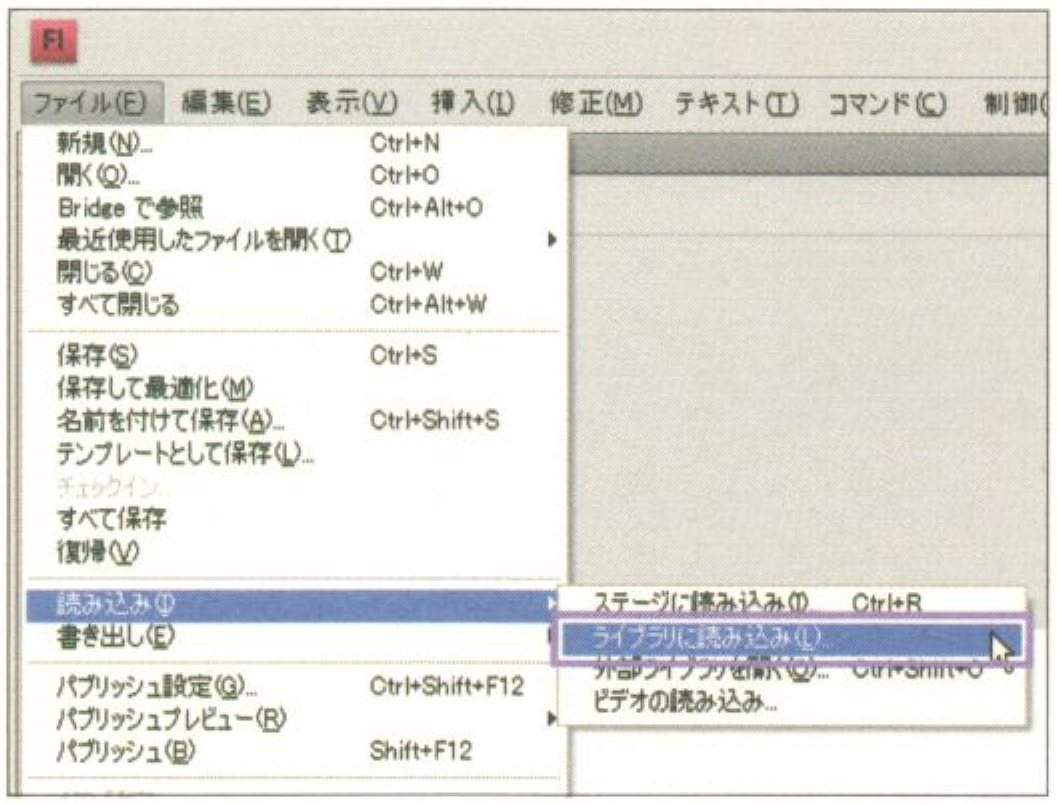
画像ファイルをFlashに読み込む

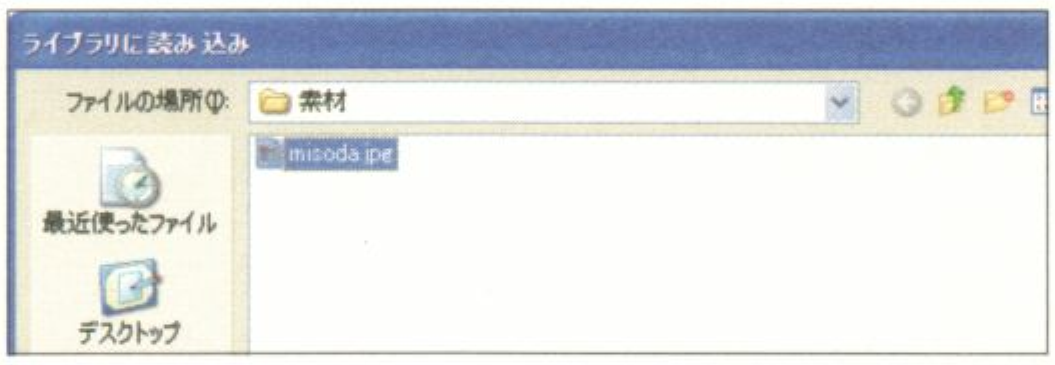
Flashファイルに画像ファイルを読み込みます。
JPEG、GIF、PNG、BMPなど、さまざまな形式のファイルを読み込めます。

▶▶ ライブラリに画像を読み込む

Flashファイルに画像ファイルを読み込む際には、［ファイル→読み込み→ライブラリに読み込み］を実行します。表示されるダイアログで、読み込みたい画像ファイルを選択し、「開く」ボタンを押します。なお、［ファイル→読み込み→ステージに読み込み］を実行すると、読み込んだ画像データが自動的にステージに配置されます。

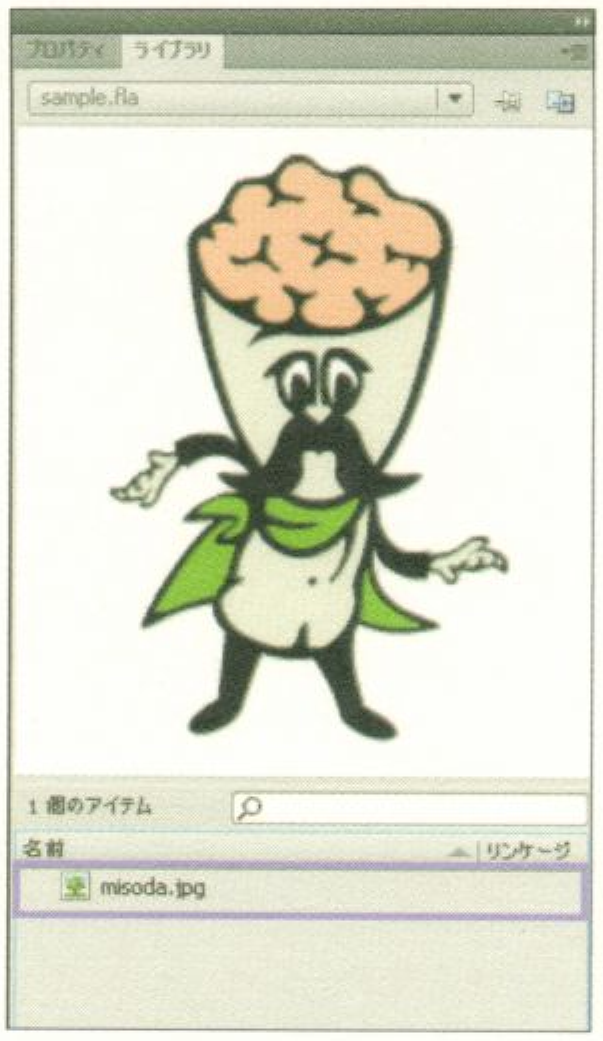
Flashでは、「JPEG」や「GIF」、「PNG」、「BMP」など、さまざまなファイル形式の画像を読み込むことができます。一般的に透過情報が含まれている画像は「PNG」形式のものを利用することが多いです。





参考：Flash CS4でサポートされているベクターファイルまたはビットマップファイルの形式
http://help.adobe.com/ja_JP/Flash/10.0_UsingFlash/WSd60f23110762d6b883b18f10cb1fe1af6-7ea8a.html#WSd60f23110762d6b883b18f10cb1fe1af6-7ea3a

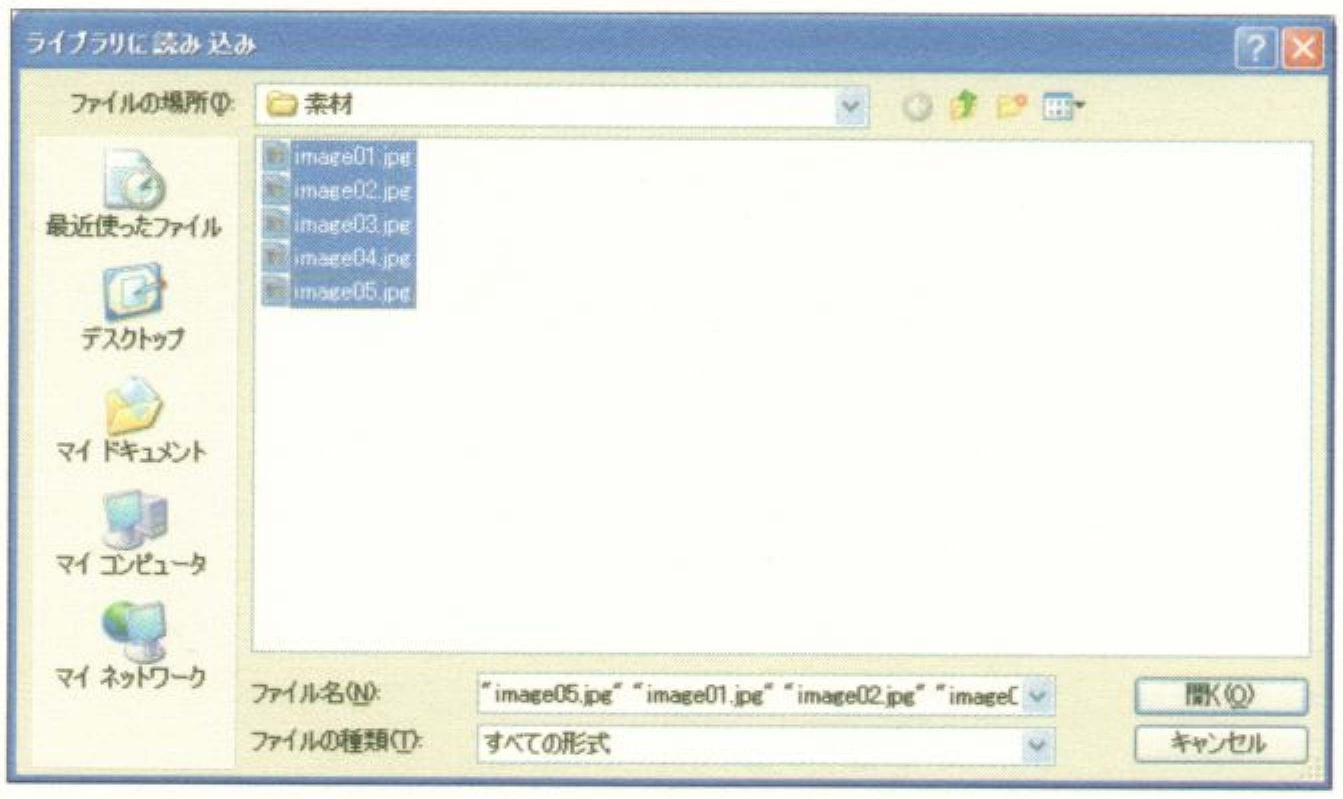
読み込まれた画像ファイルはライブラリ内に格納されます。
画像アイテムを使用したいときは、ライブラリからドラッグすることで画像をステージ上に配置することができます。



ライブラリに格納された画像ファイルには、アイコンがファイル名の前に表示される

▶▶ 複数のファイルを同時に読み込む

複数のファイルを読み込みたいときには、読み込みたい画像ファイルを選択する際に、Shiftキーを押しながら複数のファイルを選択します。この状態で「開く」ボタンを押すことで、複数の画像ファイルを一度に読み込むことができます。

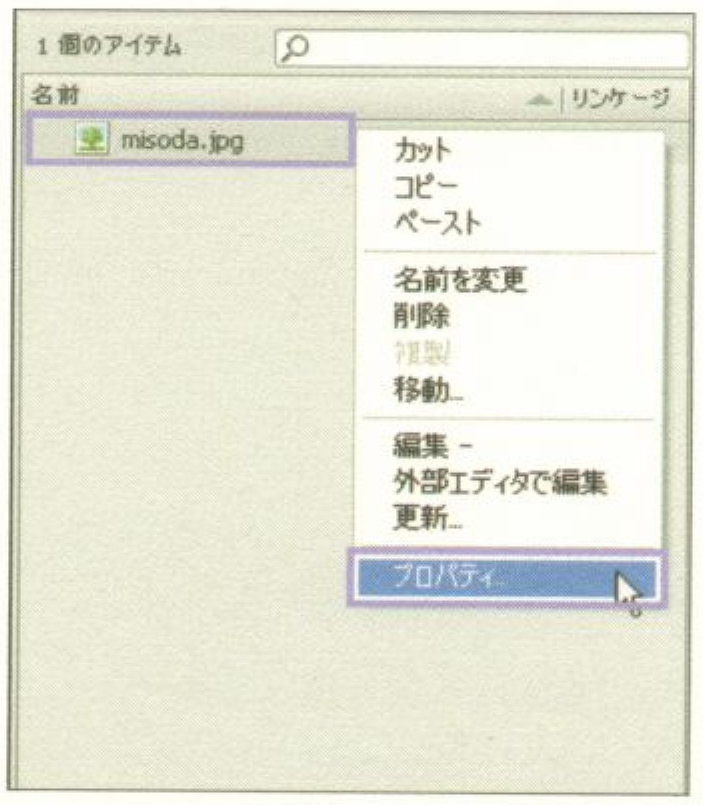


読み込んだ画像の画質を変更する

ライブラリパネルから、読み込んだ画像ファイルの画質を変更することができます。

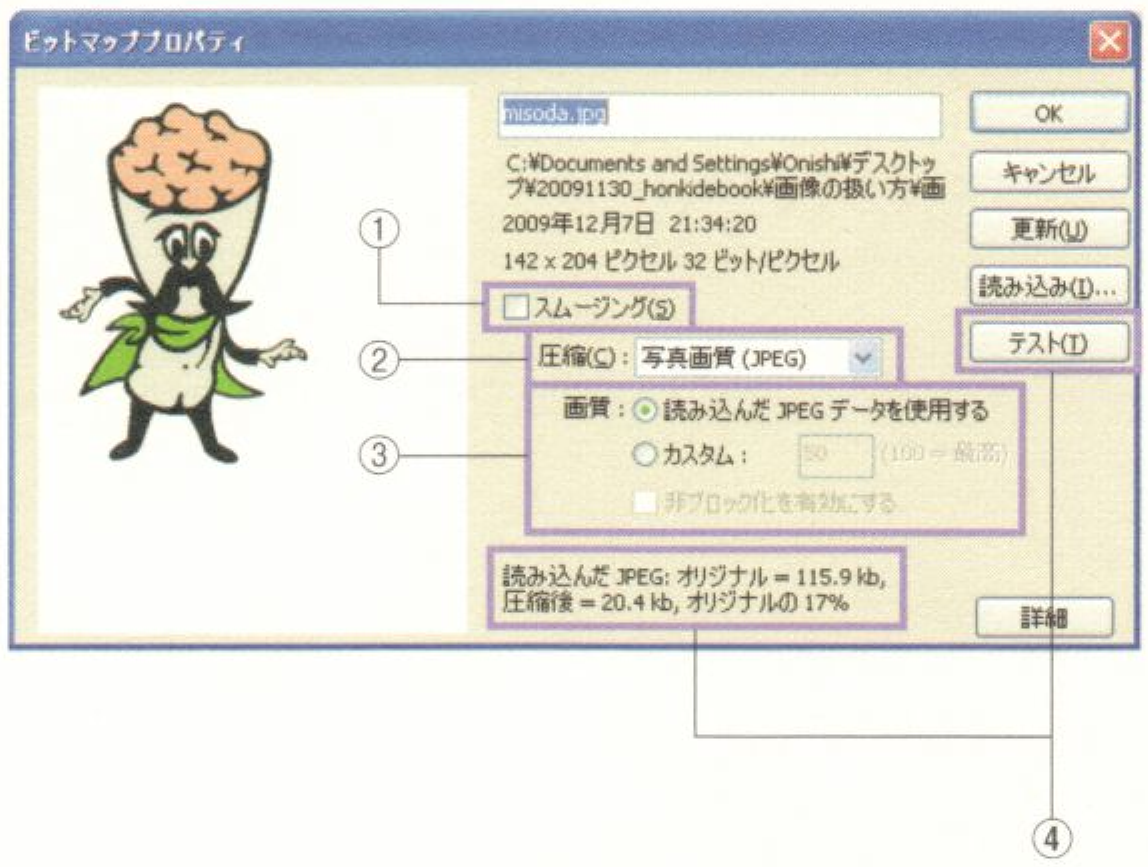
▶▶ 画像の画質を変更する

画質を変更したいライブラリの画像アイテムを選択して、右クリック→［プロパティ...］を選択します。
次ページの［ビットマッププロパティ］ウィンドウが開くので、ここで画質の設定を行います。各項目の設定を行い、「OK」ボタンを選択すると設定が保存されます。



ライブラリパネルで右クリック→［プロパティ...］を選ぶ

ここで設定できる項目や、使用できる機能には以下があります。



①スムージング

画像にスムージングをかけるか設定します。この項目を選択すると拡大/縮小時の画像の品質が向上します。

②圧縮

画像の圧縮方式を選択します。選択可能な圧縮方式は以下の2つです。

写真画質（JPEG）	設定した画質の値に合わせて画像データが圧縮されます。数字が大きいほど書き出した際のファイルサイズも大きくなります。
劣化なし（PNG/GIF）	画像がロスレス圧縮され、データが劣化しません。

③画質

圧縮で「写真画質（JPEG）」を選択しているときのみ設定できます。「読み込んだJPEGデータを使用する」を選択するともともとの画質になります。「カスタム」を選択した場合は、1～100までの任意の数値を設定することが可能です。

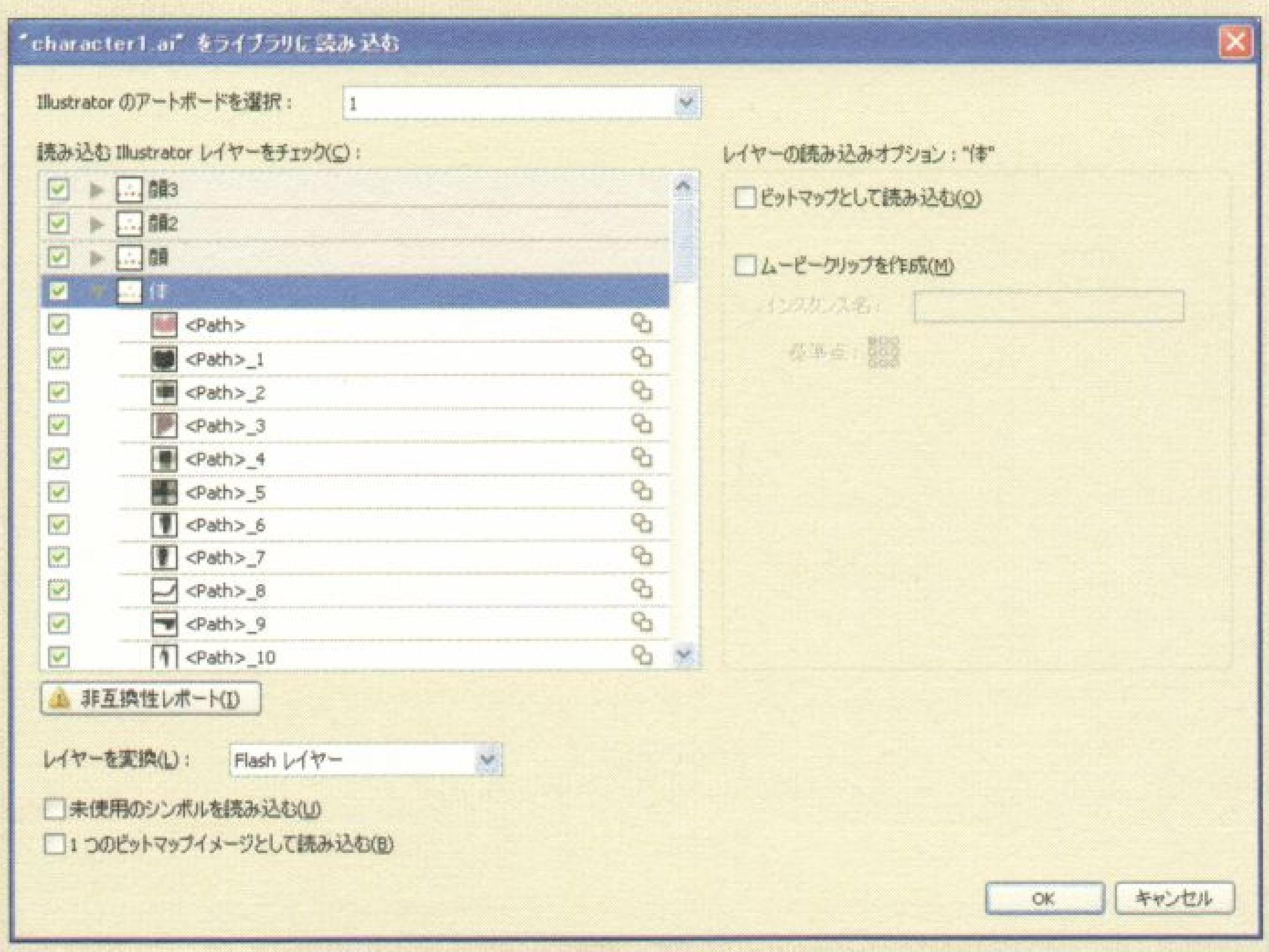
④テストボタン

各項目を設定し、このボタンを押すことで、圧縮後のファイルサイズを表示することができます。

COLUMN

「.aiデータ、.psdデータ」の読み込み

ai形式やpsd形式のデータを読み込む際にも、通常の画像と同様に「ライブラリに読み込み」を実行します。この場合、読み込み項目を詳細に設定できるウィンドウが立ち上がります。このウィンドウ上では、レイヤーやオブジェクトごとに読み込むかどうかの選択を行えたり、読み込み形式をビットマップ形式かベクター形式かを選択したりと、非常に詳細な読み込み設定が行えます。



LECTURE 2-4

Flashによる表現

サウンドの扱い方

Flashムービーには音を入れ込むことができます。ここでは、音の取り込み方から、音の再生方法までを解説します。また、それ以外に音をActionScriptで制御するやり方もありますが、詳しくはLECTURE 5-7で解説しています。

合わせて読むと便利な項目

■ActionScriptでのサウンドの制御……………………………………………………P151へ

サウンドを扱う

Flashムービー上でサウンドを再生する方法は、「flaファイルに取り込んでタイムライン上で再生」「flaファイルに取り込んだファイルをActionScriptを使って再生」「外部のサウンドファイルを再生」の3つがあります。

3つの再生方法には、それぞれ次のようなメリットとデメリットがあります。

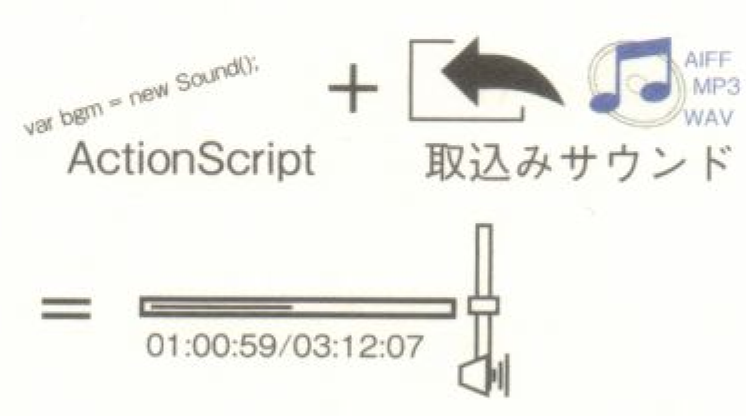
flaファイルに取り込んで、タイムライン上で再生

メリット	・タイムラインに同期して再生させるなど、アニメーションとの相性がとても良いのが特徴です。 ・Flashだけで、簡単な編集（サウンドファイルの一部を利用するなど）ができます。
デメリット	・flaファイルに取り込むことでファイルサイズが大きくなります。 ・再生の停止や、リプレイなどサイトBGMのような再生制御には不向きです。
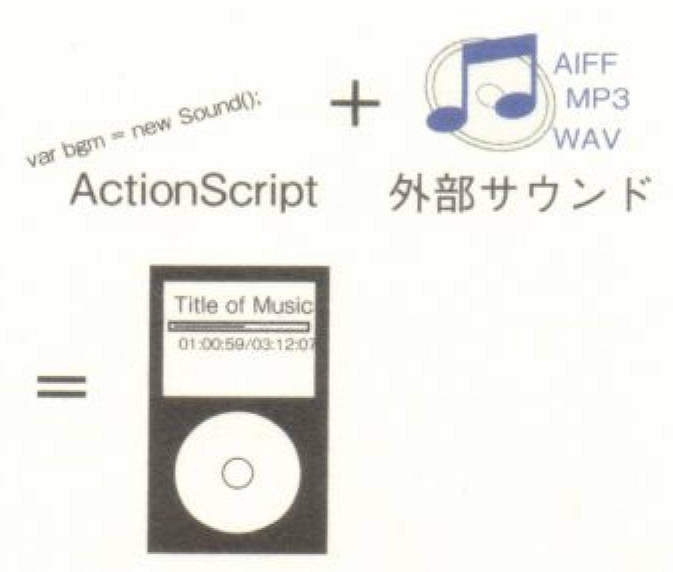


flaファイルに取り込んだサウンドファイルをActionScriptを使って再生

メリット	・再生の停止や、リプレイなど、サイトBGMのような再生制御に向いています。 ・サウンドのボリューム制御などが可能です。
デメリット	・flaファイルに取り込むことでファイルサイズが大きくなります。



外部のサウンドファイルを再生

メリット	・再生の停止や、リプレイなどサイトBGMのような再生制御に向いています。 ・読み込みが自由にできるので、ファイルサイズの肥大化を抑えることができます。 ・サウンドプレイヤーのようなアプリケーションを作ることも可能です。 ・flaファイルを変更せずに、再生させるサウンドを差し替えることが容易です。 ・サウンドのボリューム制御などが可能です。
デメリット	・外部ファイルなので、読み込むまでの時間の管理など、タイミングの制御をプログラムで作る必要があります。 ・読み込み不可の形式もあるので、サウンドファイルの形式に注意が必要です。



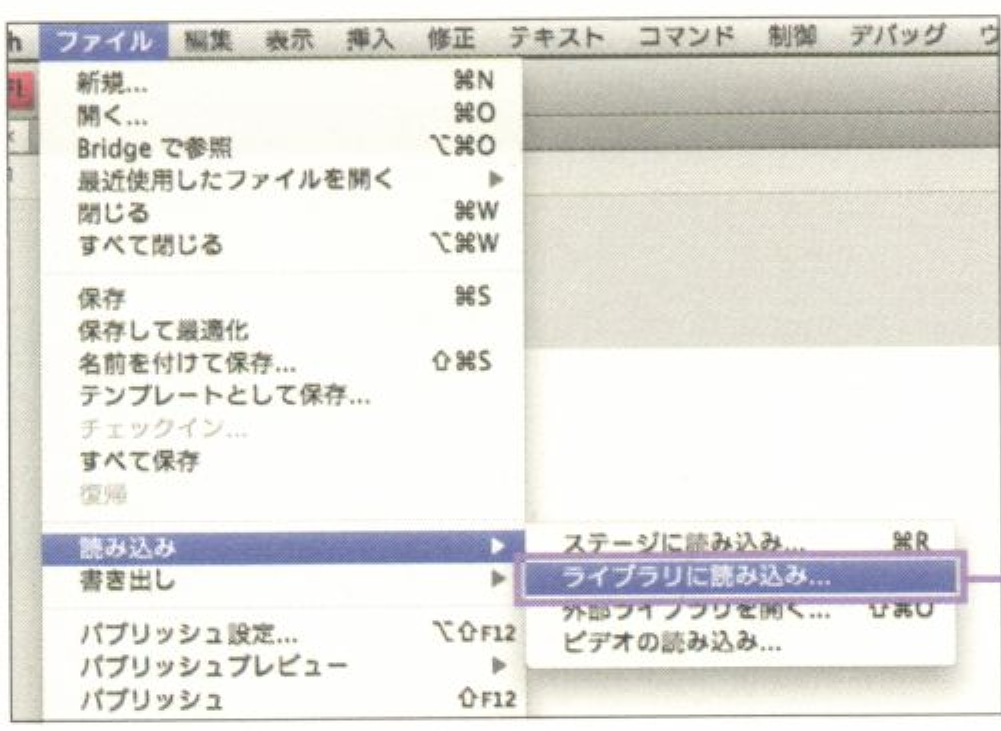
サウンドファイルの取り込みと再生方法

3つの方法のうちの1番目、「flaファイルに取り込んで、タイムライン上で再生」について、
具体的な方法と設定を解説していきます。

▶▶ 音の取り込み方

Flashで音ファイルを読み込む場合は以下の手順で行います。

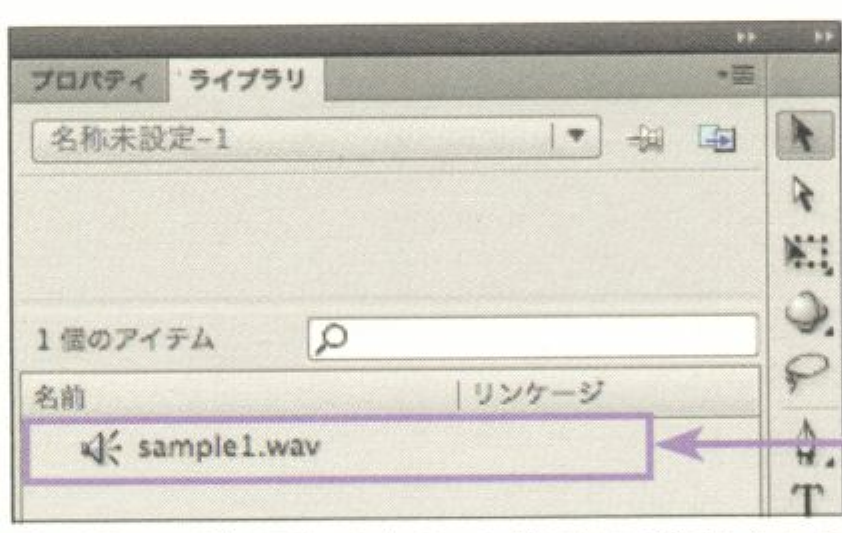
1. [ファイル→読み込み→ライブラリに読み込み]を選択します。
2. [読み込み]ダイアログボックスで、必要なサウンドファイルを指定して開きます。読み込みが完了するとライブラリに追加されています。



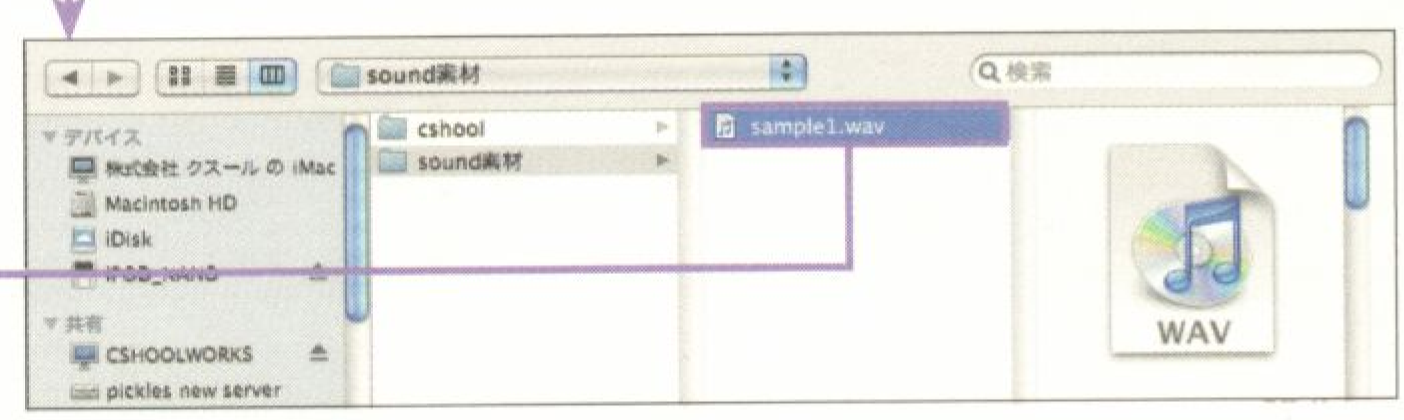
取り込み可能な音声ファイルフォーマット

形式	Windows	Macintosh
ASND（Adobe® Soundbooth™ のネイティブサウンド形式）	○	○
WAV	○	QT
AIFF	QT	○
MP3	○	○
Sound Designer® II	×	QT
サウンドのみのQuickTimeムービー	QT	QT
Sun AU	QT	QT
System 7 サウンド	×	QT

※QT：QuickTime® 4 以降がインストールされている場合

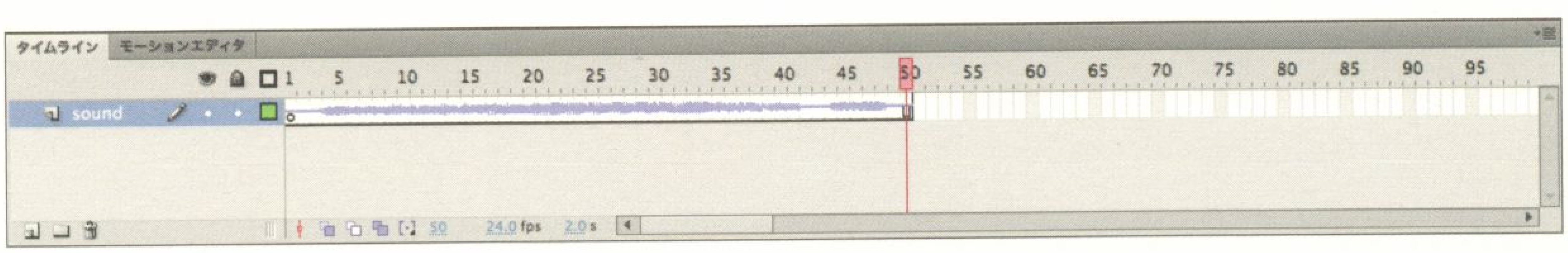


読み込みが完了するとライブラリに追加される



▶▶ タイムラインへのサウンドの追加

1. サウンド専用のレイヤーを追加して、再生したい音のファイルをライブラリからステージ上へドラッグ＆ドロップします。タイムラインを再生することで、音が鳴ります。

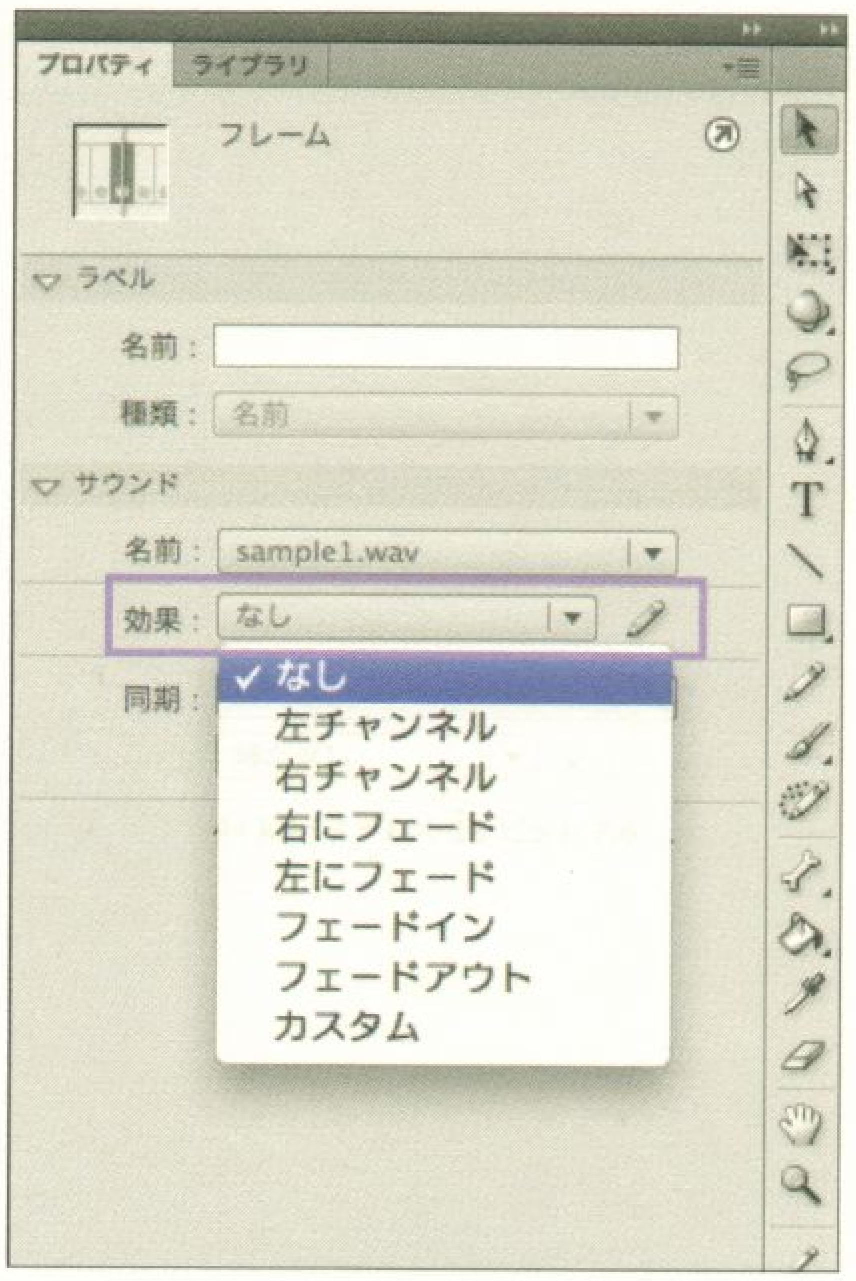


複数のサウンドをタイムラインで再生したい場合は、サウンドごとに新しいレイヤーを追加しましょう。
レイヤーは別れていますが、最終的にミキシングされて再生されます。

再生の設定

効果

タイムラインに適用した音には、効果を付けることができます。
サウンドを入れたレイヤーをクリックして、[プロパティ] パネルで「効果」のプルダウンを選択します。

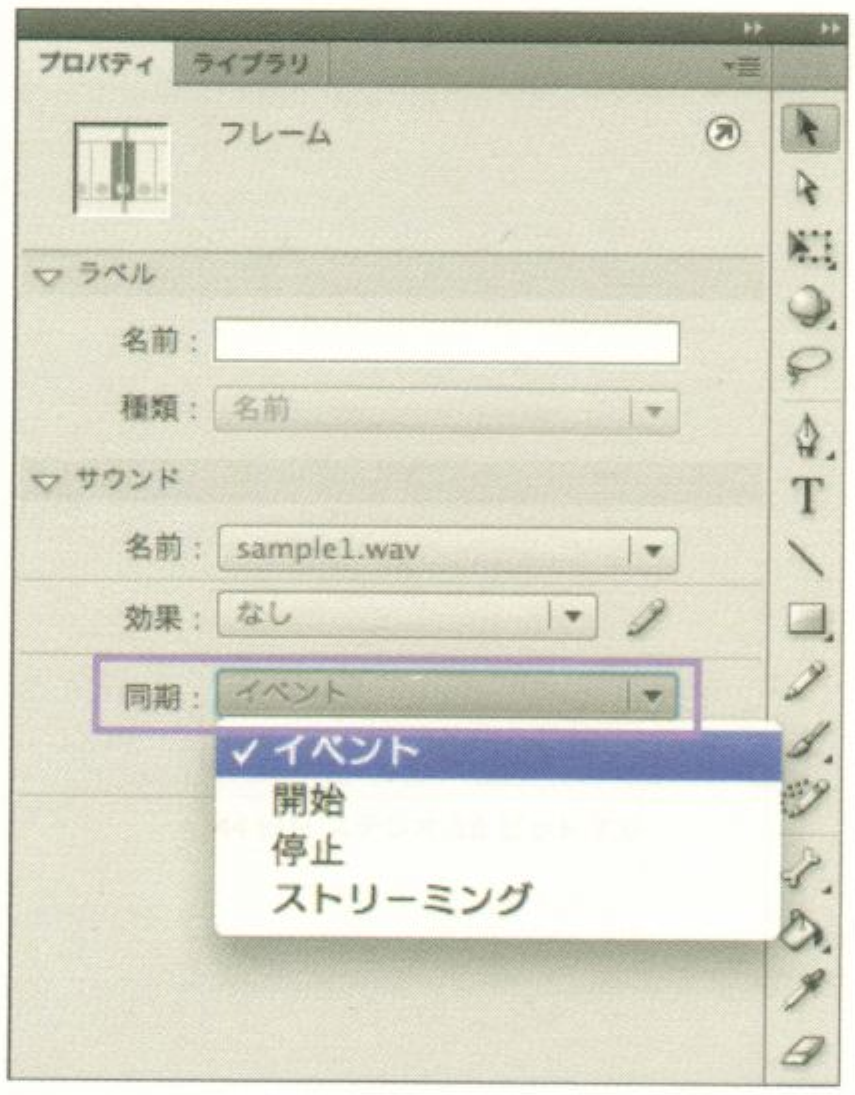


効果の種類

なし	サウンドファイルにどの効果も適用されません。
左チャンネル	左チャンネルだけでサウンドが再生されます。
右チャンネル	右チャンネルだけでサウンドが再生されます。
左へフェード	左のチャンネルから右のチャンネルにサウンドがシフトします。
右へフェード	右のチャンネルから左のチャンネルにサウンドがシフトします。
フェードイン	サウンドが継続している間、サウンドのボリュームが徐々に大きくなります。
フェードアウト	サウンドが継続している間、サウンドのボリュームが徐々に小さくなります。
カスタム	[エンベロープの編集] を使用してサウンドのインポイントとアウトポイントを独自に作成できます。

同期

1. タイムラインで、サウンドファイルを含んでいる最初のフレームを選択します。
2. [プロパティ] パネルで「同期」から以下4つのいずれかを選択します。



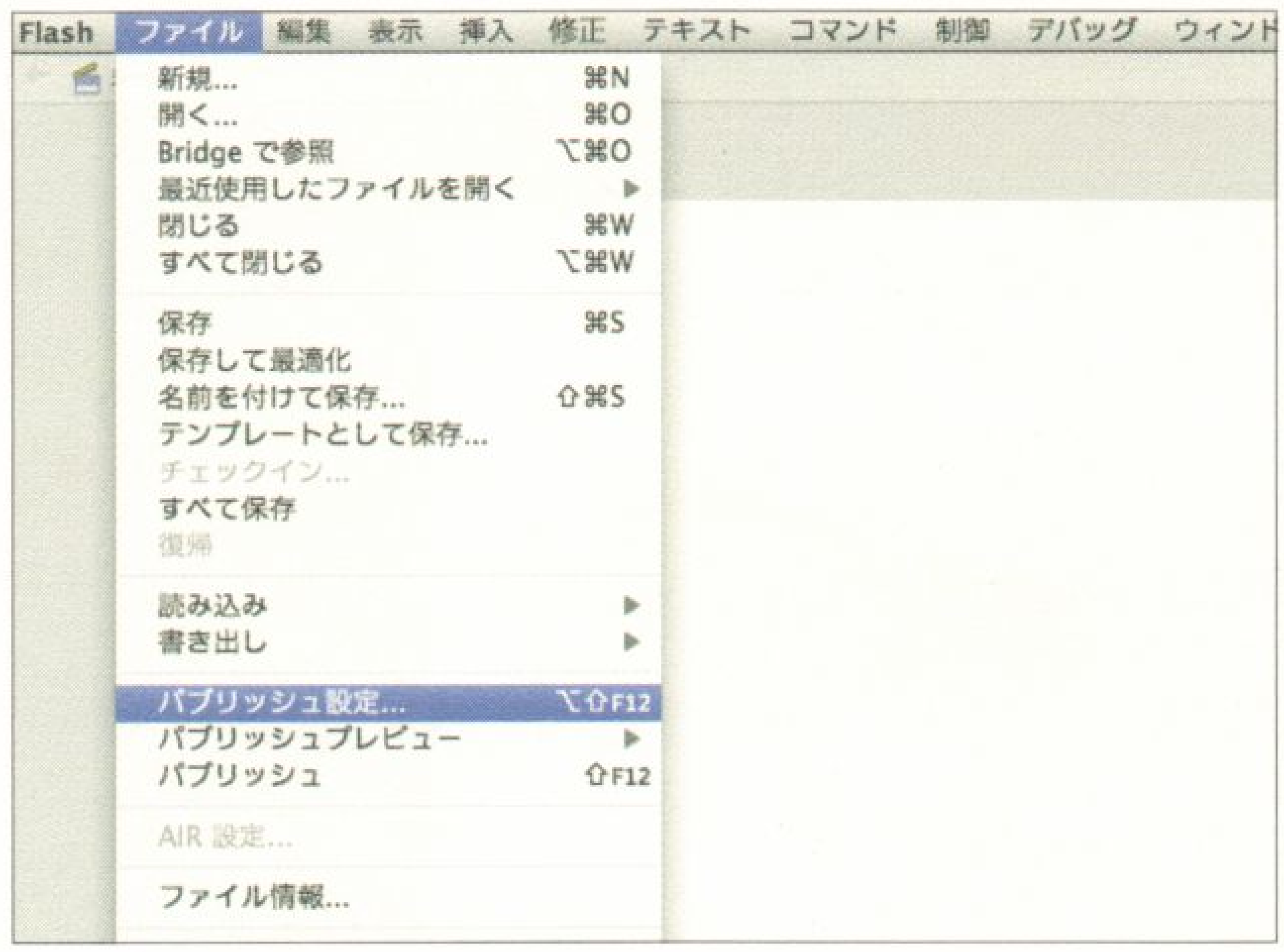
再生方法の種類

イベント	サウンドの入っているタイムラインの先頭が再生されると設定されるサウンドです。
ストリーミング	ナレーションなどアニメーションに同期したい場合に用いられます。
開始	再生を開始する際に用いられます。停止と対で覚えましょう。
停止	指定したサウンドがこの「停止」を指定したフレームで停止します。

▶▶ 書き出しの設定

サウンドの品質

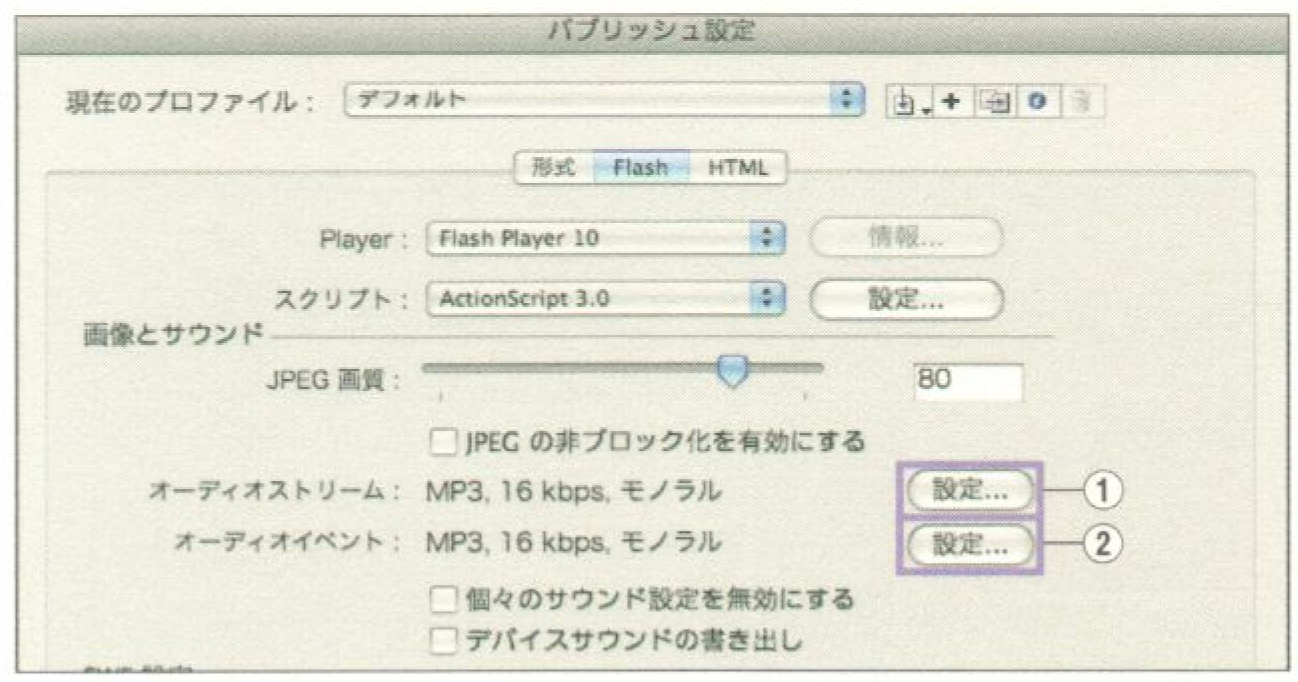
Flashムービーを書き出す（ムービープレビューをする）と、音質が低下する場合があります。それは、Flashの書き出し設定に原因があります。Flashムービーの音質を上げるには、［ファイル→パブリッシュ設定］を開きます。



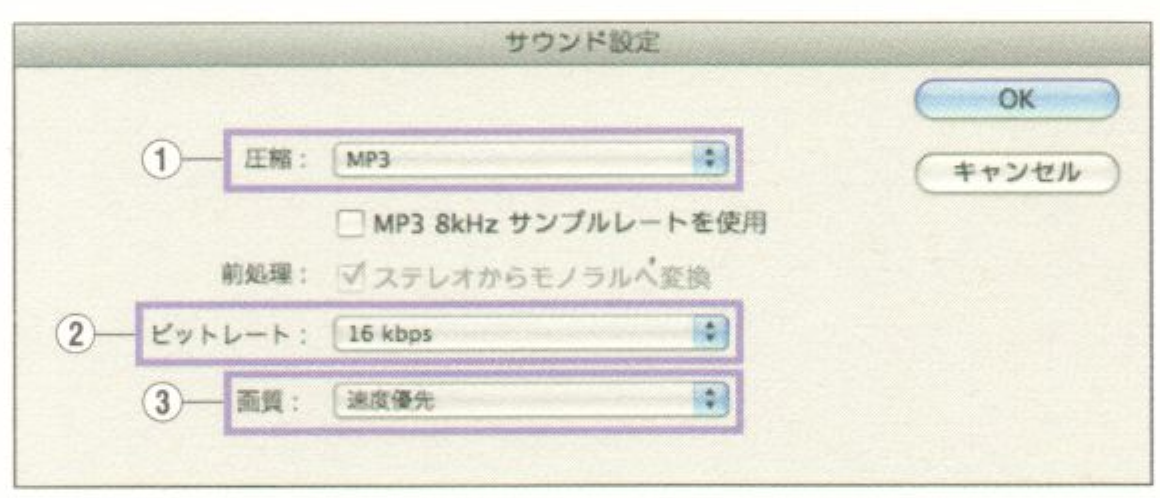
音質の設定には、オーディオストリームとオーディオイベントの2種類があります。

オーディオストリーム	再生方法でストリーミングに設定された音の品質
オーディオイベント	再生方法でイベントに設定された音の品質

［パブリッシュ設定］の「Flash」タブの中の「オーディオストリーム」①、または「オーディオイベント」②を選択し、音質を設定します。



設定を選択すると［サウンド設定］ウィンドウが立ち上がります。
ここで、重要な3点の設定を紹介します。



①圧縮
　サウンドファイルの圧縮形式を決めます。

②ビットレート
　書き出すサウンドファイルのビットレート（ビット/秒）を決めます。
　初期設定では、16kbpsになっているが、音質をムービープレビューで確認しながら決めます。128kbpsまで上げるとかなり高音質であるが、ファイルサイズが大きくなるので注意が必要です。

③画質
　圧縮速度と音質を決めます。

圧縮の種類

無効	圧縮を無効にします。
ADPCM	8ビットから16ビットのサウンドデータ用の形式です。ボタンクリックのようなイベントサウンドに最適です。
MP3	高音質で高圧縮です。
Raw	非圧縮にします（ファイルサイズの増加注意）。
Speech	ナレーションなど話言葉に最適です。

画質の種類

速度優先	圧縮は速くなりますが、音質は低下します。
標準	圧縮は多少遅くなりますが、音質は向上します。
品質優先	圧縮は最も遅くなりますが、音質は最も高くなります。

LECTURE 2-5

Flashによる表現

映像の扱い方

Flashでは映像ファイルを読み込んで、再生・制御を行うことが可能です。ここでは、映像ファイルを読み込む方法や種類について説明していきます。

映像が使われているサンプル

■ PRACTICE 7：
外部SWFを読み込み、制御する … P265へ

合わせて読むと便利な項目

■ フレームとタイムライン…………… P038へ
■ Loader ……………………………… P158へ

Flashでの映像の扱い方

映像を読み込んでSWF上で再生するには、次の3つの方法があります。

Flashドキュメントへのビデオの埋め込み

映像ファイルをFlashファイルに埋め込んで制御することが可能です。埋め込んだ映像ファイルはムービークリップと同様にタイムラインの制御を行うことができます。
ただし、映像ファイルそのものをファイル内に埋め込むので、SWFのファイルサイズが大きくなります。

Webサーバーからのビデオのプログレッシブダウンロード

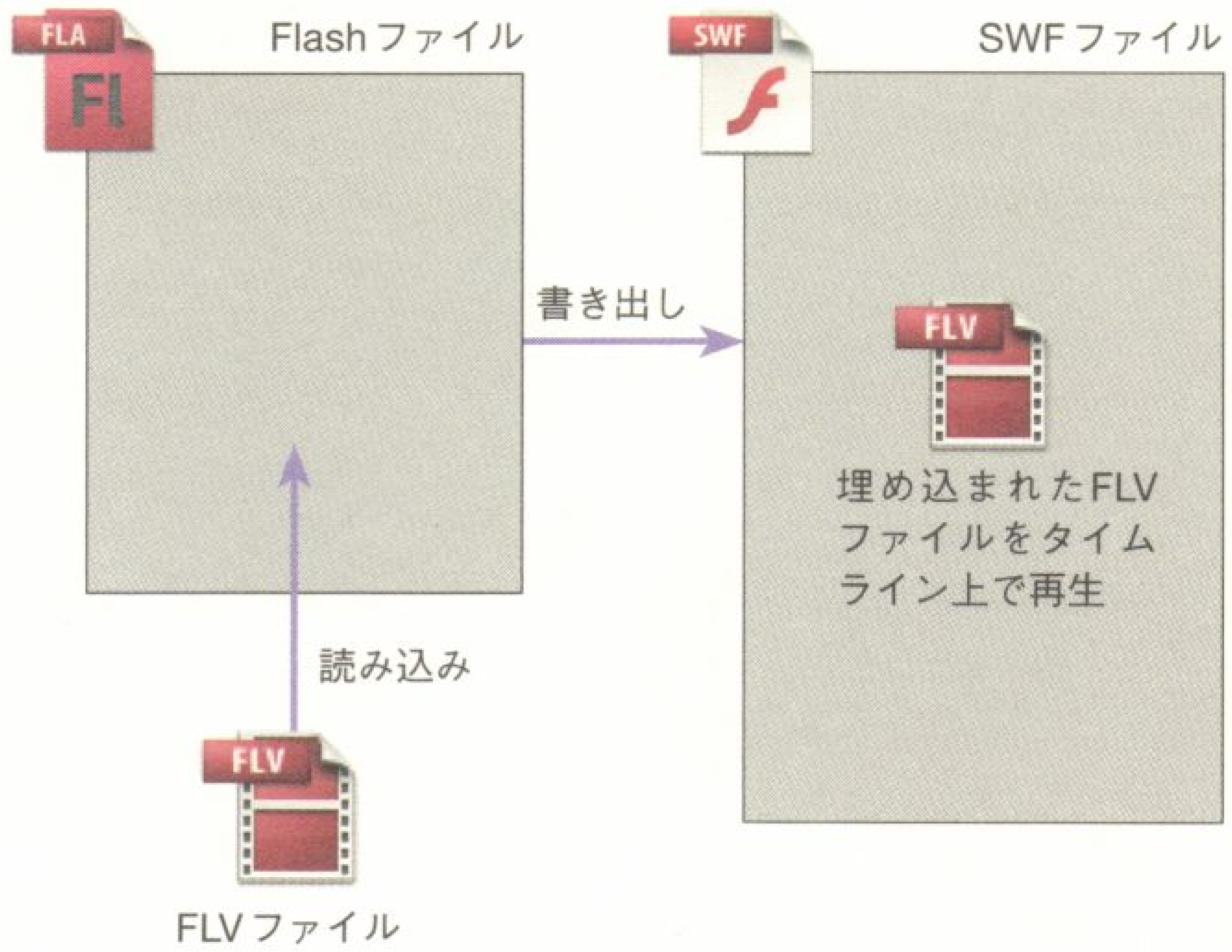
SWFファイル外に置かれたビデオファイルを動的に読み込んで再生します。SWF内にビデオファイルが含まれないためファイル容量が軽減され、パブリッシュに要する時間も短くできます。

Adobe Flash Media Serverによるビデオのストリーミング

Adobe Flash Media Serverを用いて、ビデオのストリーミング配信を行うことができます。
SWF内にビデオファイルが含まれないためファイル容量が軽減され、パブリッシュに要する時間も短くできます。またこの方式では、ファイルがユーザーのPCにキャッシュされないため、ビデオコンテンツの保護に適しています。

▶▶ Flashに取り込んでフレームに配置

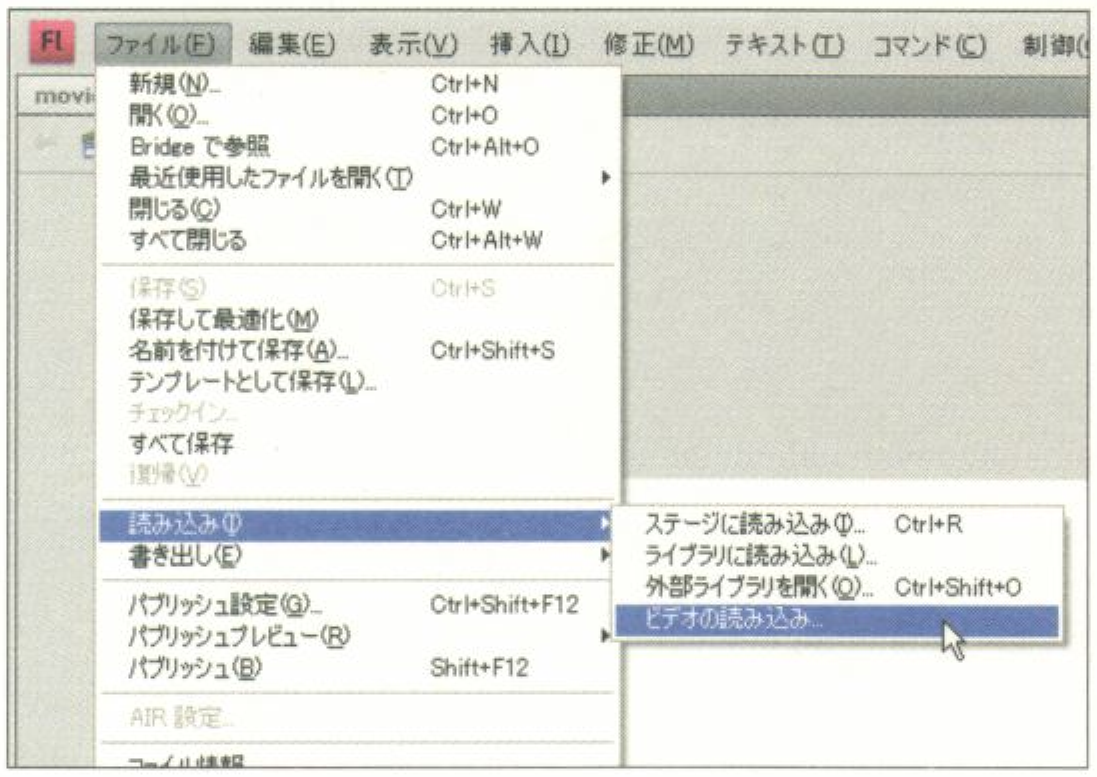
映像ファイルをFlashファイルに読み込み、タイムライン上で再生する方法を解説します。



手順 1

Flashで、[ファイル→読み込み→ビデオの読み込み]を実行します。[ビデオの読み込み]ウィザードが起動します。

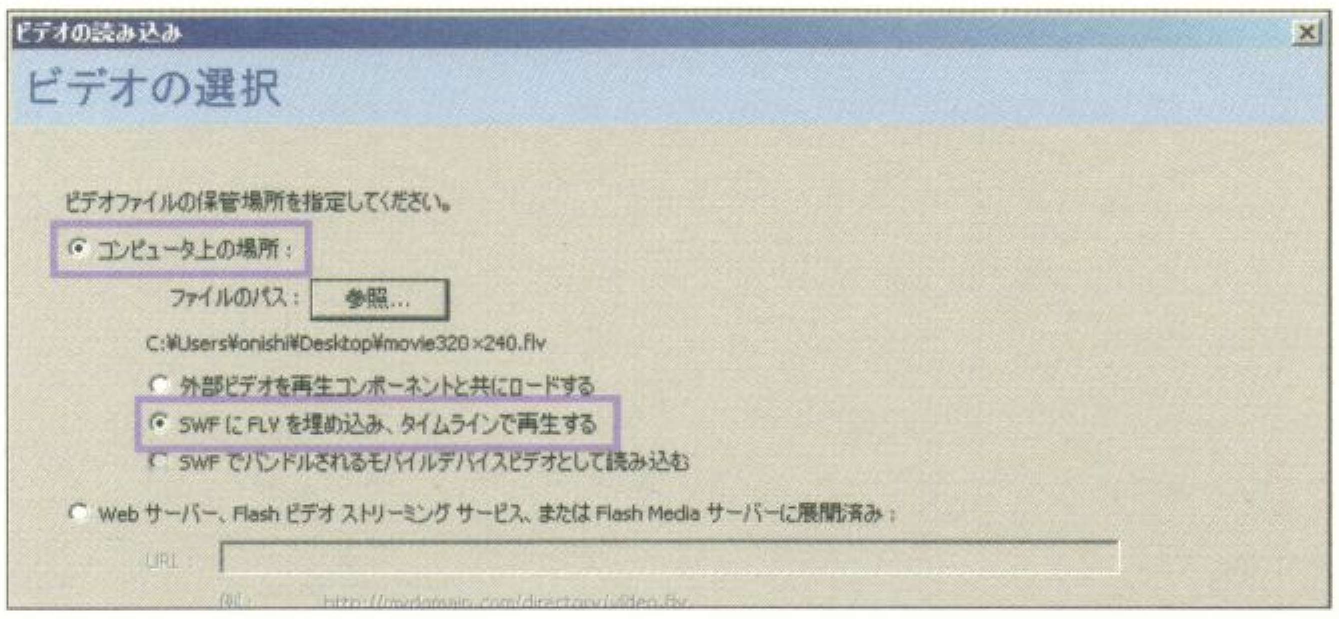
ビデオの読み込みの実行



手順 2

ビデオファイルの保管場所を指定します。
今回はPC上にあるファイルを読み込んで使いたいので、「コンピュータ上の場所」を選択します。「参照」ボタンを押して、読み込みたいファイルの保管場所を指定します。
ビデオの読み込みの方法を「SWFにFLVを埋め込み、タイムラインで再生する」に指定します。

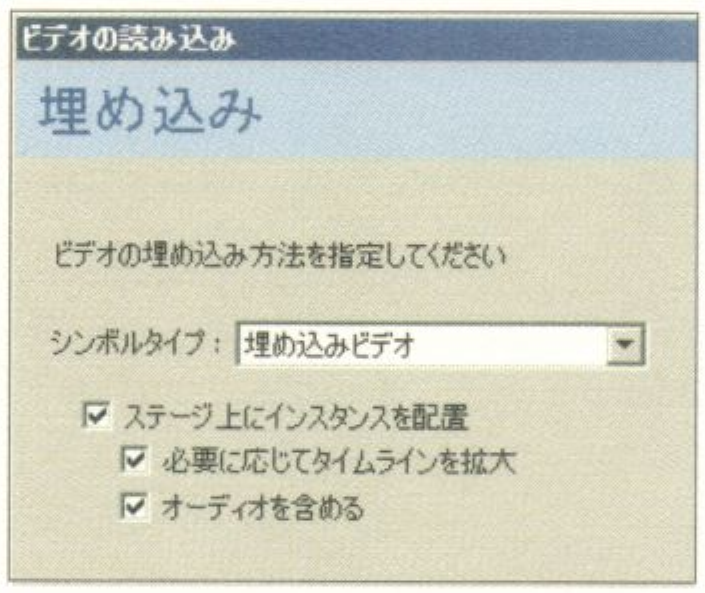
ビデオの読み込みウィザード



手順 3

ビデオの埋め込み方法を指定します。各項目はそれぞれ以下のようになっています。

ビデオの埋め込み方法

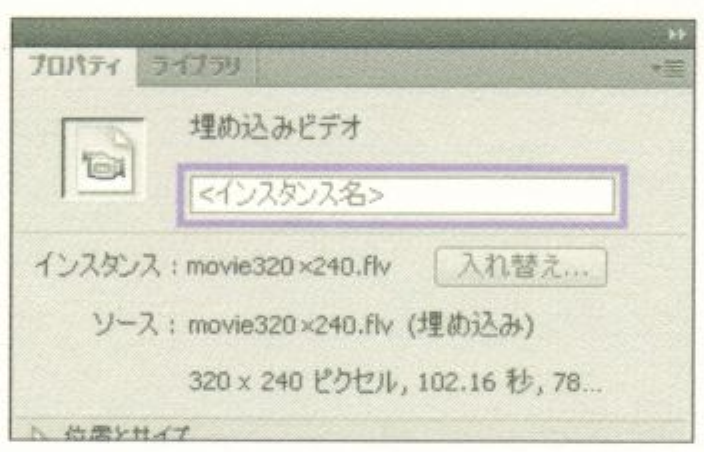


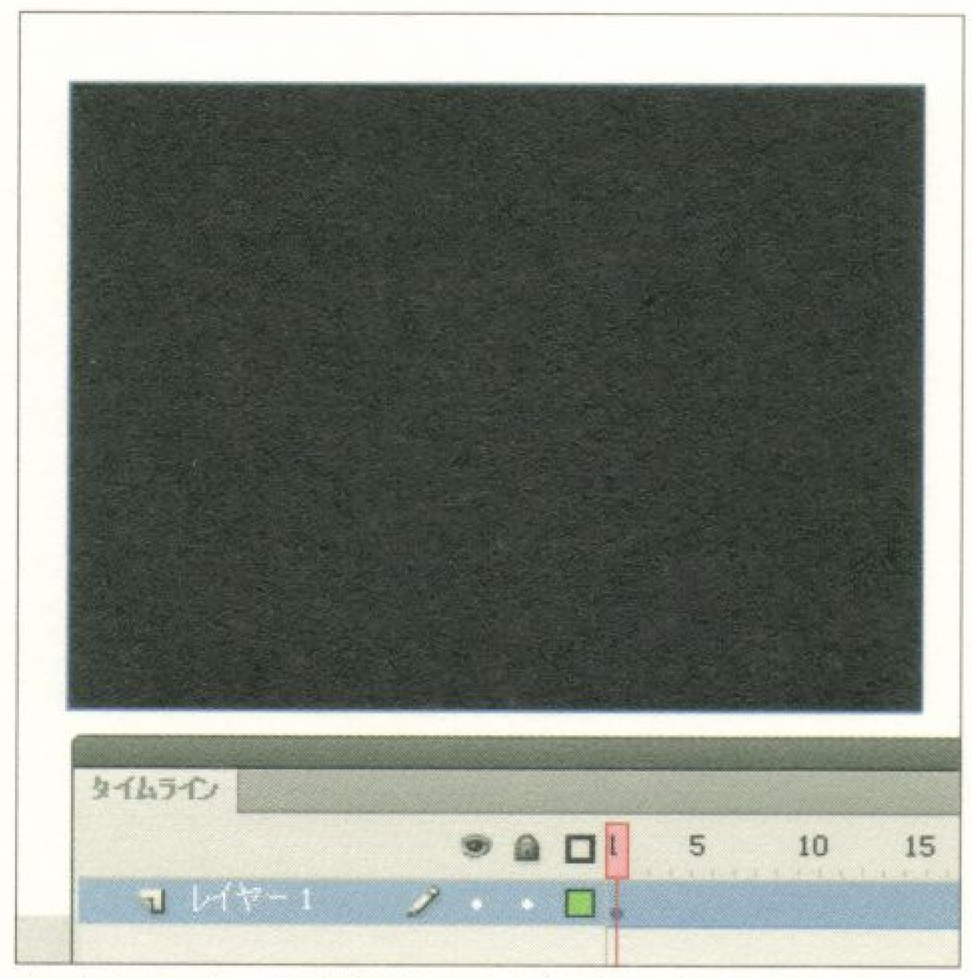
シンボルタイプ	ビデオをどのシンボルタイプで埋め込むか指定します。選択可能なシンボルタイプは「埋め込みビデオ」「ムービークリップ」「グラフィック」の3種類です。
ステージ上にインスタンスを配置	この項目を選択すると、読み込み後自動的にステージ上にビデオが配置されます。
必要に応じてタイムラインを拡大	この項目を選択すると、ステージ上にビデオが配置された際にステージ上のタイムラインが短かった場合、自動的にタイムラインをビデオの長さに拡大します。
オーディオを含める	この項目を選択すると、ステージに配置されたビデオにオーディオが含まれます。

手順 4

手順3で「ステージ上にインスタンスを配置」にチェックを入れていると、タイムラインにビデオが配置されます。

配置されたビデオはムービークリップなどと同様に、タイムラインの操作や画面上への配置を行うことができます。



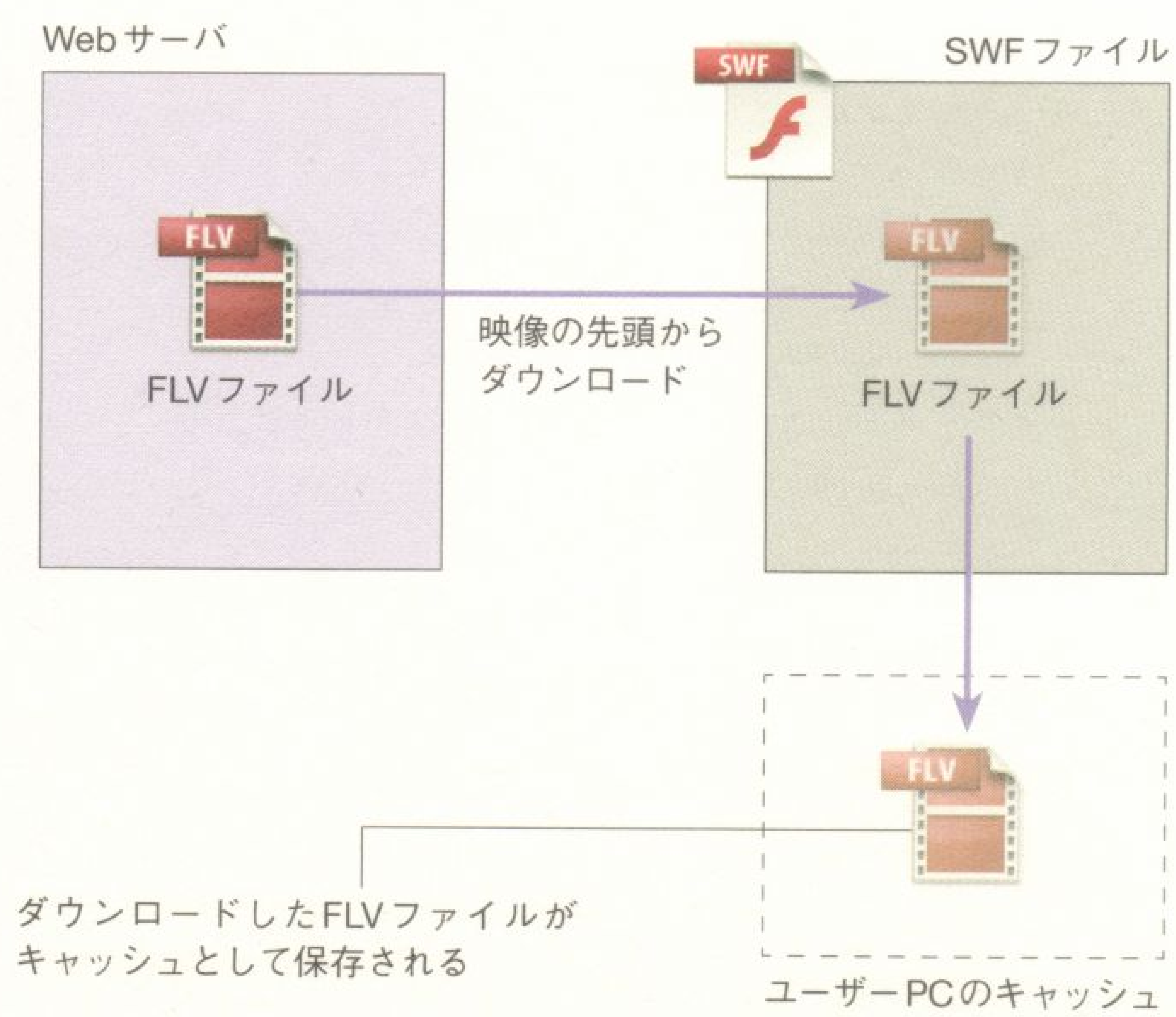


タイムラインに配置されたビデオインスタンス

▶▶ Webサーバーからのビデオのプログレッシブダウンロード

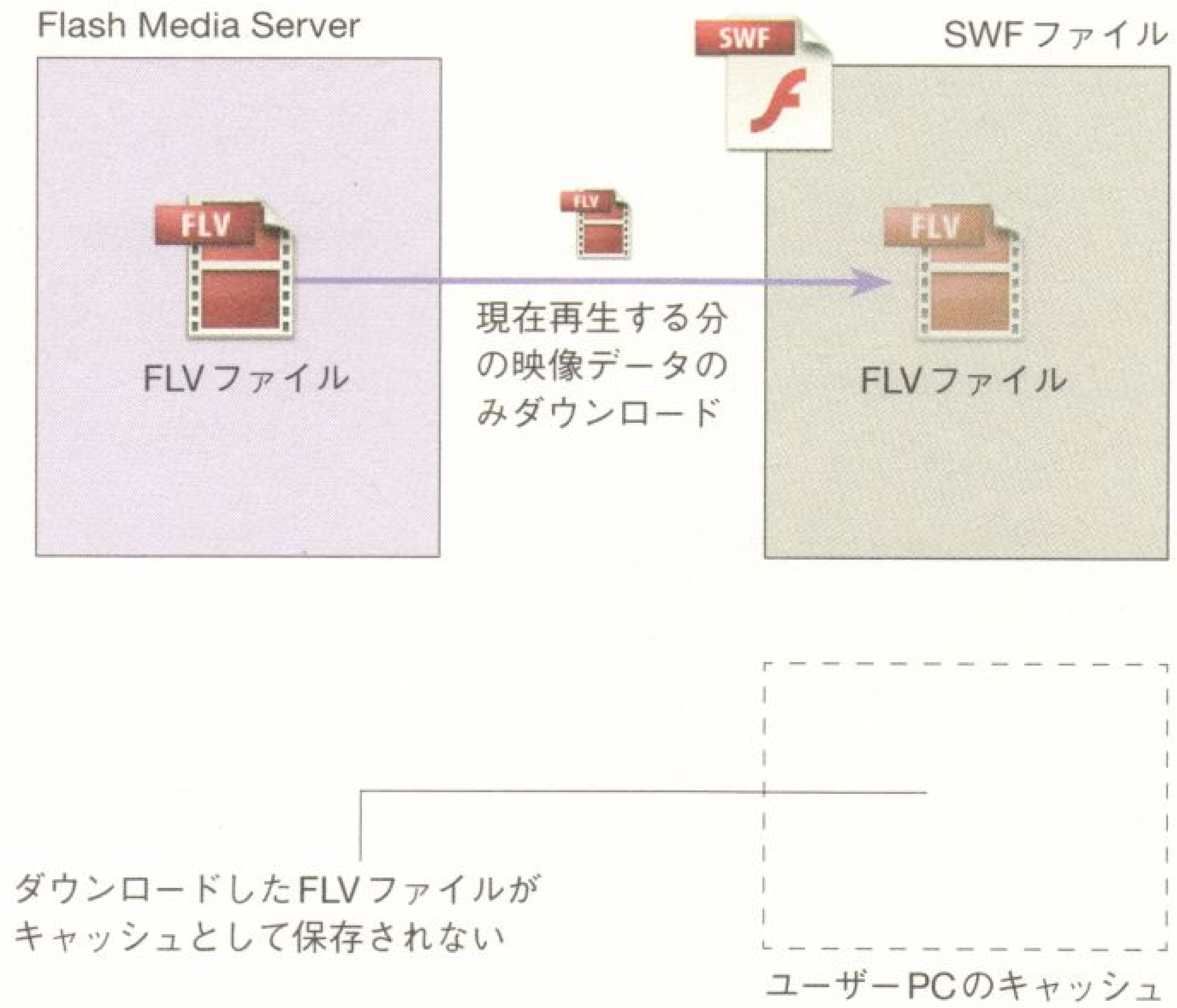
Webサーバ上など、SWFファイル外に置かれたビデオファイルを動的に読み込んで再生します。外部のFLVファイル、F4VファイルをFLVPlaybackコンポーネントやActionScriptを使って、読み込みやビデオの制御を行うことが可能です。

この方法では、ファイルの読み込みを進めながらビデオの再生が可能なので、ビデオ開始までのユーザーの待ち時間を大幅に短縮することができます。



▶▶ Adobe Flash Media Serverによるビデオのストリーミング

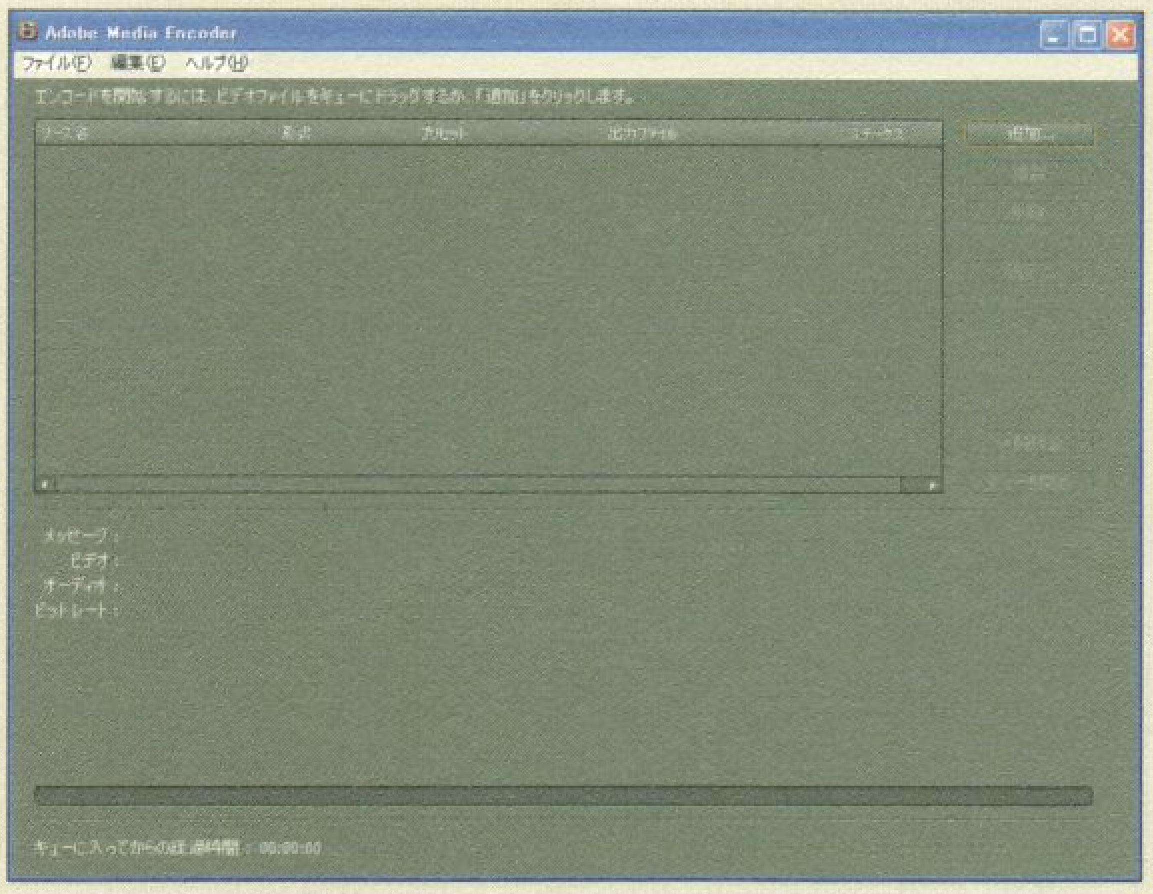
Flash Media Serverを利用してビデオのストリーミングを行う場合は、通常のWebサーバからのプログレッシブダウンロードの方式より、沢山の利点があります。
ユーザーがビデオのダウンロードに使用する回線をFlash Media Server上で制御できるため、ユーザの帯域幅の動的な振り分けや同時接続数の設定、複数のユーザーでストリーミングを共有することも可能です。
また、Flash Media Serverを用いたストリーミングでは、ユーザーのPCにファイルがキャッシュされないため、コンテンツのセキュリティや保護にも適しています。



COLUMN

Adobe Media Encoderについて

選択したファイルがFLV形式でない場合は、「Adobe Media Encoderを起動する」ボタンからAdobe Media Encoderを起動し、映像ファイルをFLV形式に変換する必要があります。Adobe Media Encoderで読み込みが可能なファイル形式は以下のURLより確認できます。
Adobe Media Encoder CS4の読み込みでサポートされているファイル形式 http://help.adobe.com/ja_JP/AdobeMediaEncoder/4.0/WS8A54CAA7-268E-4af0-B8A4-0EA3EACEC1A3.html



Adobe Media Encoder

LECTURE 3-1

ActionScriptの考え方

ActionScriptとは

ActionScriptは、Flash、Flex、AIRによるコンテンツやアプリケーションでユーザーの操作やデータの処理などを可能とする、Adobe Flash PlayerおよびAdobe AIRのランタイム環境用のプログラム言語です。ActionScriptを使用することで、タイムラインによるアニメーションなど一方向性のコンテンツ制作だけでなく、ユーザーのマウスやキー操作のインタラクションの感知や、複雑なデータ処理、ウェブカメラやプリンタなどの外部デバイスの制御を行うことができます。

ActionScriptのバージョン

Flashは1996年のリリース以降、プログラム言語のアップデートを重ねています。
ここでは各ActionScriptのバージョンごとの特徴や歴史について解説します。

各バージョンの特徴

ActionScriptには「1.0」、「2.0」、「3.0」の3つのバージョンがあり、それぞれ右表のような特徴があります。
ActionScript 1.0と2.0はAVM1（ActionScript Virtual Machine1）の環境で再生されるため互換性がありますが、ActionScript 3.0では再生環境がAVM2のため下位互換性はありません。

1.0	2.0	3.0
AVM1で動作		AVM2で動作
・Flash Player 5以降 ・プロトタイプベースの開発	・Flash Player 7以降 ・クラスベースの開発が可能 ・厳密なデータ型の指定が可能（コンパイル時のみ）	・Flash Player 9以降 ・クラスベースの開発が可能 ・厳密なデータ型の指定が可能（コンパイル、ランタイム時で機能） ・ドキュメントクラスの指定が可能

ActionScriptの歴史

Flashはもともとアニメーション作成ソフトに端を発した経緯もあり、Flash4以前のプログラミング言語は「アクション」と呼ばれる独自のスクリプトでした。Flash 5以降からは、JavaScriptなどの他のプログラム言語に近く、より柔軟なプログラミングが行える「ActionScript」が搭載されました。
その後Flash MX 2004において、それまでの「ActionScript」をベースにした「ActionScript 2.0」が搭載されました。
「ActionScript 2.0」は、JavaScriptなどをベースにしたプログラミング言語の規格「ECMAScript」に準拠していて、これにより本格的なWebアプリケーションの開発が可能になりました。また現在の「ActoinScript 3.0」では、オブジェクト指向がさらに強化され、大規模で複雑なアプリケーションの開発にも対応できるようになりました。

年代	Flash	Flash Player	ActionScript
1996	Flash 1	Flash Player	
1997	Flash 2	Flash Player 2	「アクション」搭載
1998	Flash 3	Flash Player 3	
1999	Flash 4	Flash Player 4	
2000	Flash 5	Flash Player 5	「ActionScript」搭載
2002	Flash MX	Flash Player 6	
2003	Flash MX 2004	Flash Player 7	「ActionScript 2.0」搭載
2005	Flash 8	Flash Player 8	
2007	Flash CS3	Flash Player 9	「ActionScript 3.0」搭載
2008	Flash CS4	Flash Player 10	

ActionScript 3.0で開発する利点

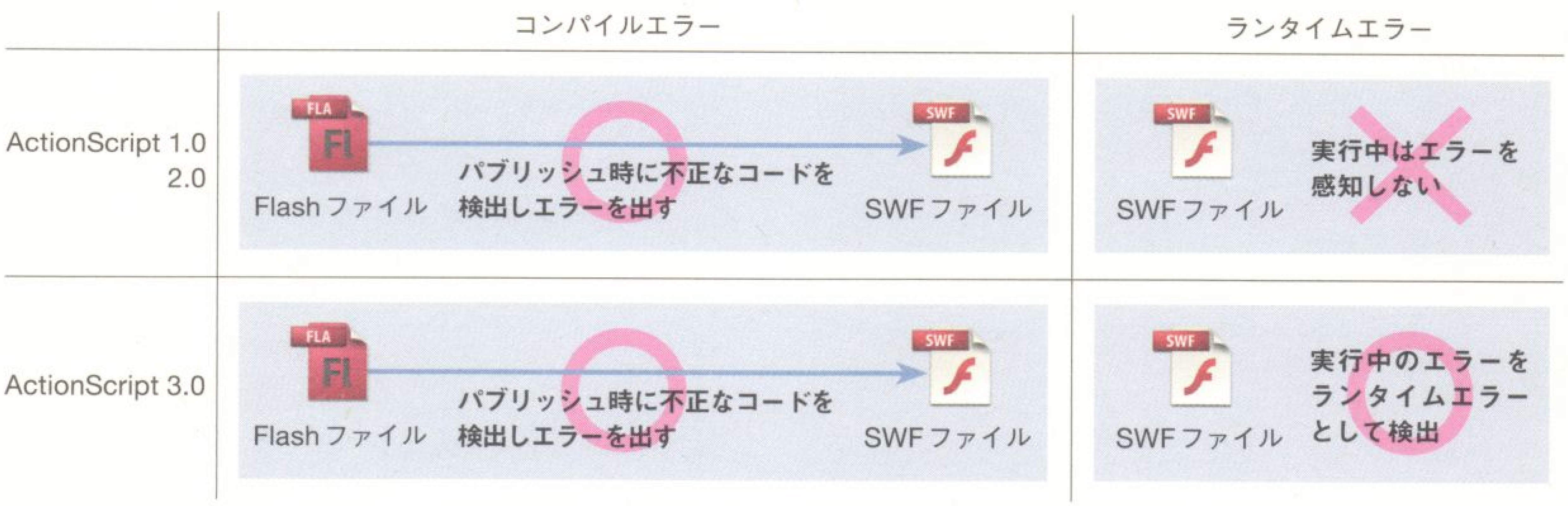
ActionScript 3.0でのプログラミングは、
処理速度の向上やデバッグの効率化などさまざまなメリットが得られます。

▶▶ 処理速度が速い

FlashPlayer 9以降に搭載されているActionScript 3.0の再生環境AVM2（ActionScript Virtual Machine2）では、FlashPlayer 8以前の再生環境に比べ飛躍的に処理速度が向上しました。

▶▶ 効率的なデバッグが可能に

ActionScript 2.0以前では、コンパイル時のエラーを捕捉することはできても、Flashコンテンツ再生中に発生するランタイムエラーを捕捉することができませんでした。ActionScript 3.0では、開発者がランタイムエラーを捕捉することができるため、非常に効率的にデバッグ作業を行うことができるようになっています。



▶▶ オブジェクト指向プログラミングに適したプログラミング体系

ActionScript 3.0は、ECMAScript 標準第4版として提案された「ECMA-262 Edition 4」に準拠しており、非常に柔軟かつ効率的なプログラミングが可能になっています。これにより、大規模で複雑なアプリケーションや、複数人で行うアプリケーションなどの開発が非常に行いやすくなりました。

COLUMN

オブジェクト指向とは

「オブジェクト指向」とは、プログラミングをする際の考え方や作法の1つです。いろいろな機能や情報を個別のまとまりとして持たせるため、プログラミングの再利用が可能になる、コードの可読性があがる、バグが発見しやすいなどのメリットがあります。また、個別に機能をまとめるのではなく、プログラムの処理の流れに沿って順序立てて記述していくプログラミング言語を「手続き型言語」と呼びます。

LECTURE 3-2

ActionScriptの考え方

ActionScript 3.0の書き方

ActionScript 3.0でプログラミングを始める前に、まずはActionScript 3.0の書き方を紹介します。今後必ず必要になる項目なので、まずはここで名前やルールを理解すると良いでしょう。

合わせて読むと便利な項目

■ MovieClipを制御する……………………P083へ

ドットシンタックス

複数のムービークリップやクラスを連結する際に「.(ドット)」を使うのですが、この書き方をドットシンタックスと言います。このドットシンタックスという書き方はJavaScriptと同じです。

例

```
chara.x = 100; // キャラのX位置を100にする
```

大文字と小文字は別物として扱う

ActionScript 3.0では大文字と小文字の区別を行います。一文字でも間違えると同じものと認識されません。

例

```
var a = 1;
var A = 2;
trace(a); ⬅ [出力] パネルには「1」と表示される
trace(A); ⬅ [出力] パネルには「2」と表示される
```

大文字と小文字の使い分け方

通常クラス名は大文字からはじまり、インスタンス名は小文字から始まります。絶対そうしないと動かないと言うわけではありませんが、慣習としてそうなっています。

例1

```
var car = new Car(); ⬅ carがインスタンス名で、Carはクラス名
```

例2

```
var myCar;
⬆ 複数の単語が組み合わさるときは、2つ目の単語の最初を大文字に
```

セミコロン

プログラミングの1行が終わると「;(セミコロン)」を付けます。セミコロンはここで1つの命令が終了したという印です。これを付けなくても動きますが、他の言語には必須のものもあるので、書いておく習慣をつけましょう。

例

```
var a=1;
```

コメント

プログラミングを書いているときに、あとから見直した際、このコードが何を意味しているのかが分からなくなることがあります。その際はコードの近くにコメントを書きます。
コメントはそれ以外に意味を持たないので、日本語で書いても結構です。

例1

```
//　スラッシュ2つを付けるとコメントになります。ここでプログラミングを書いても実行されません。
```

例2

```
/*

複数のコメントを書きたい場合は
このようにして「/*」と「*/」で囲ってあげると
中身の部分がすべてコメントになります。

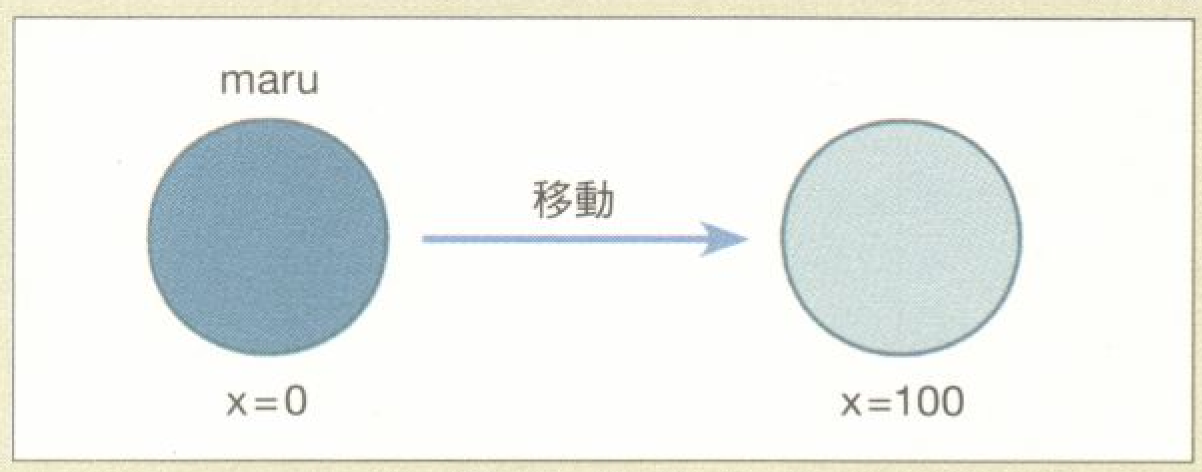
*/
```


COLUMN

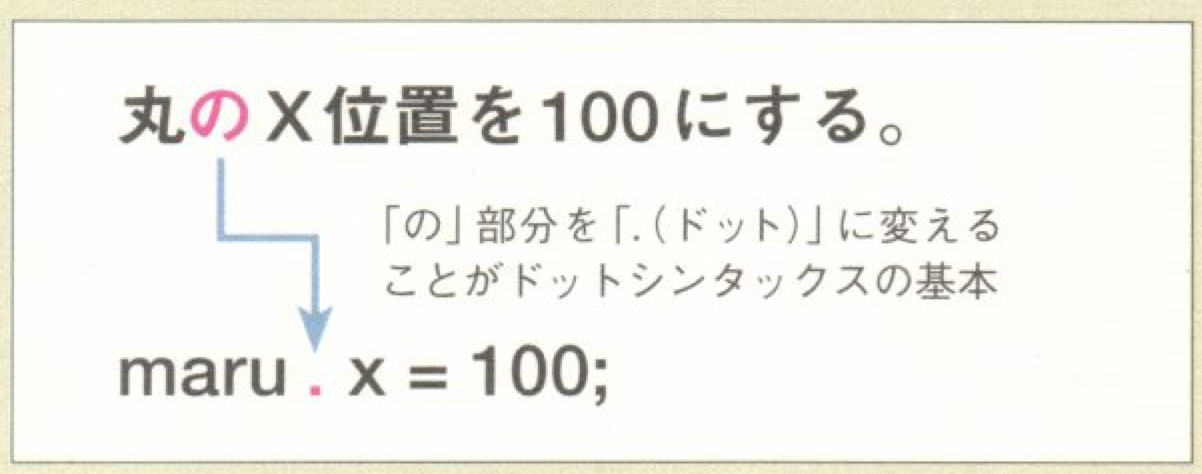
日本語でプログラミングを考える

プログラミングを勉強する際に「数学が苦手だから難しい」と言う言葉をよく聞きます。しかし、プログラミングは英語で「Programming Language」と言われているように、言語（language）の一種です。つまり国語だと思ってもらえればよいでしょう。
まずはプログラミングを考える際に、日本語から英語に翻訳するように、日本語からプログラミング言語へ置き換えるという作業をすると、プログラミングの書き方がすんなりと入ってくると思います。
プログラミングを記述する際にまずは日本語で考えます。その際にもっとも重要なのが「〜の」のあたる部分です。この「〜の」はActionScriptでは「.（ドット）」にあたります。この書き方をドットシンタックス（ドットで区切る書き方）と言います。

まずは図のように丸をActionScriptで移動させたいと考えます。日本語で考えると「丸のX位置を100にする。」です。



この「の」部分を「.（ドット）」に置き換えると考えて、プログラミングに変換すると右のようになります。



if文でも同じようなことが考えられます。「もし丸のX位置が100より大きくなったら、丸のX位置を100にする。」という文章を考えてみましょう。この場合も日本語をそのままプログラミングにすると、プログラミングの組み立てもそれほど難しく感じないことでしょう。

```
if ( maru.x > 100 ) {
⇧ もし丸のX位置が100より大きくなったら、
  maru.x = 100;
  ⇧ 丸のX位置を100にする。
}
```

LECTURE 3-3

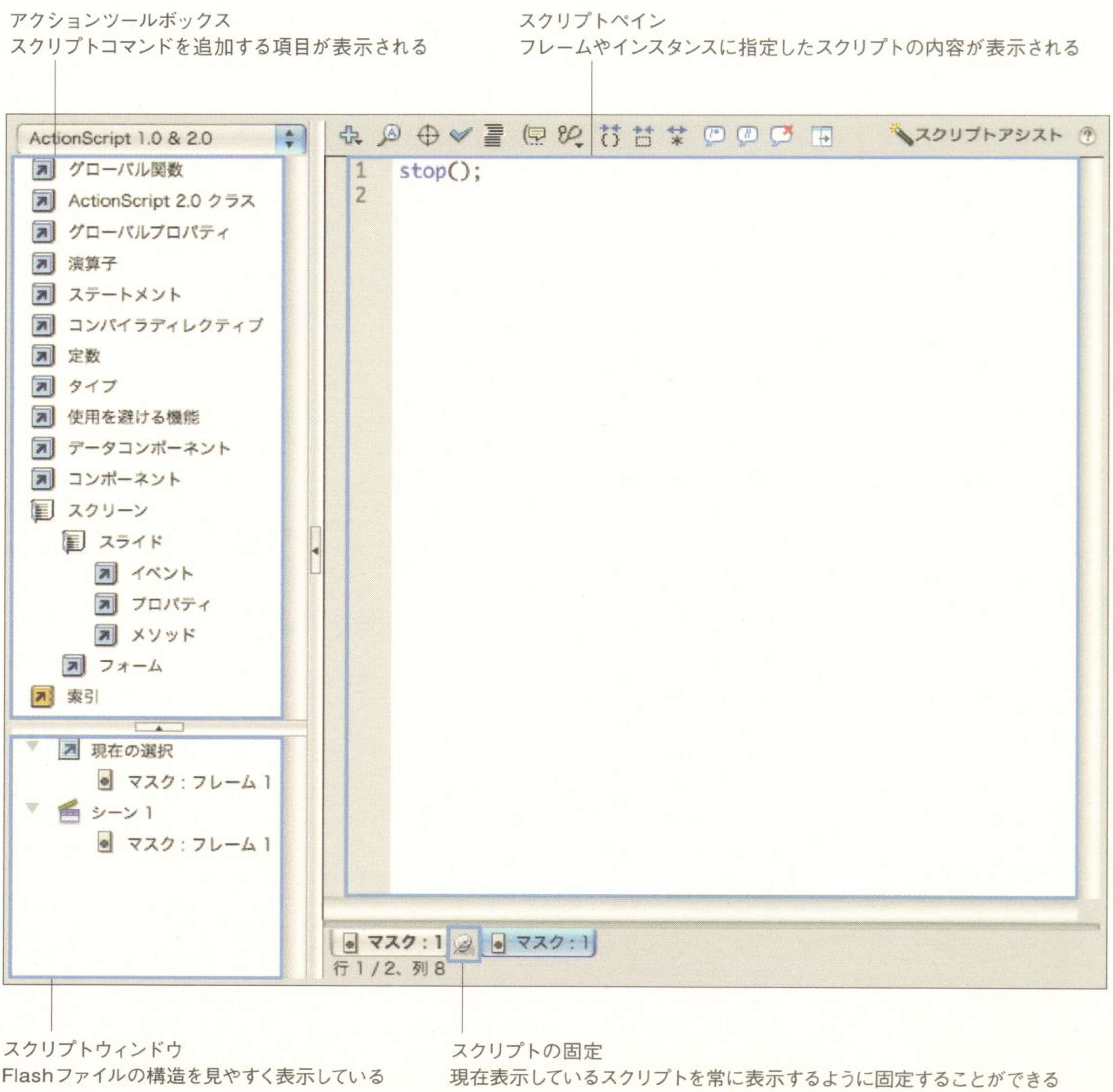
ActionScriptの考え方

アクションパネルの使い方

ActionScriptは、メインのタイムラインやムービークリップのタイムラインのキーフレームを選んで、[アクション] パネルに記述していきます。

アクションパネルの画面構成と機能説明

ここではアクションパネルのインターフェイスと機能について解説します。



▶▶ アクションパネルの機能

アクションパネルでは、コードの記述のほかに、「自動フォーマット」や「スクリプトアシスト」などの機能があります。

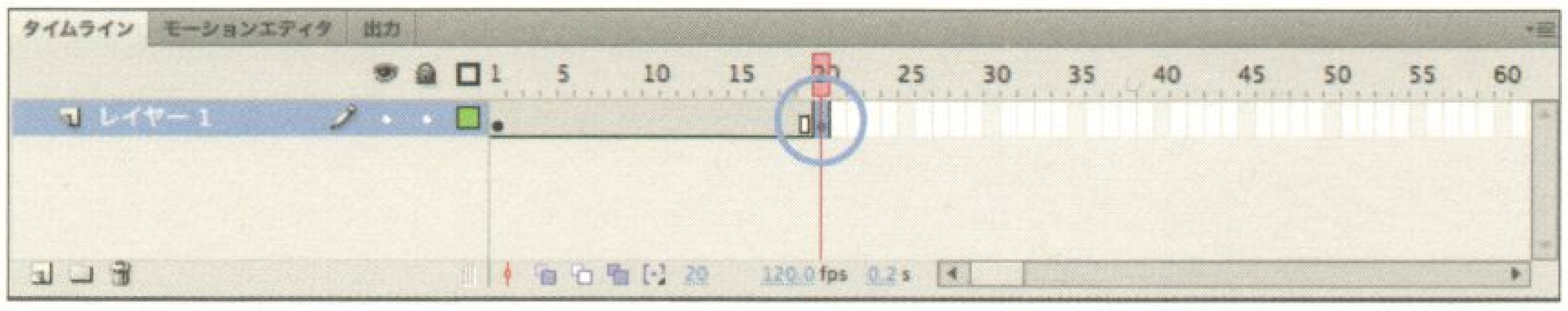
① スクリプトに新規アイテムを追加
②【置換】……………………………… スクリプト内の文字を置換する画面を表示します。
③【ターゲットパスの挿入】……… スクリプトの対象となるインスタンスを選択する画面を表示します。
④【シンタックスチェック】………… クリックすると、スクリプトのシンタックスチェックを行います。
⑤【自動フォーマット】………………… スクリプトのシンタックスに合わせて改行やインデントを行います。
⑥【コードのヒントを表示】……… 環境設定のActionScriptカテゴリで「コードヒント」がチェックされていない場合、クリックするとコードのヒントを表示します。
⑦【デバッグオプション】………… ブレークポイントの設定や削除をするメニューを表示します。
⑧【スクリプトアシスト】…………… カーソルがある行のスクリプト作成を補助する画面が表示されます。
⑨【ヘルプ】………………………… カーソルがある行に関連するコードヘルプをブラウザで表示します。

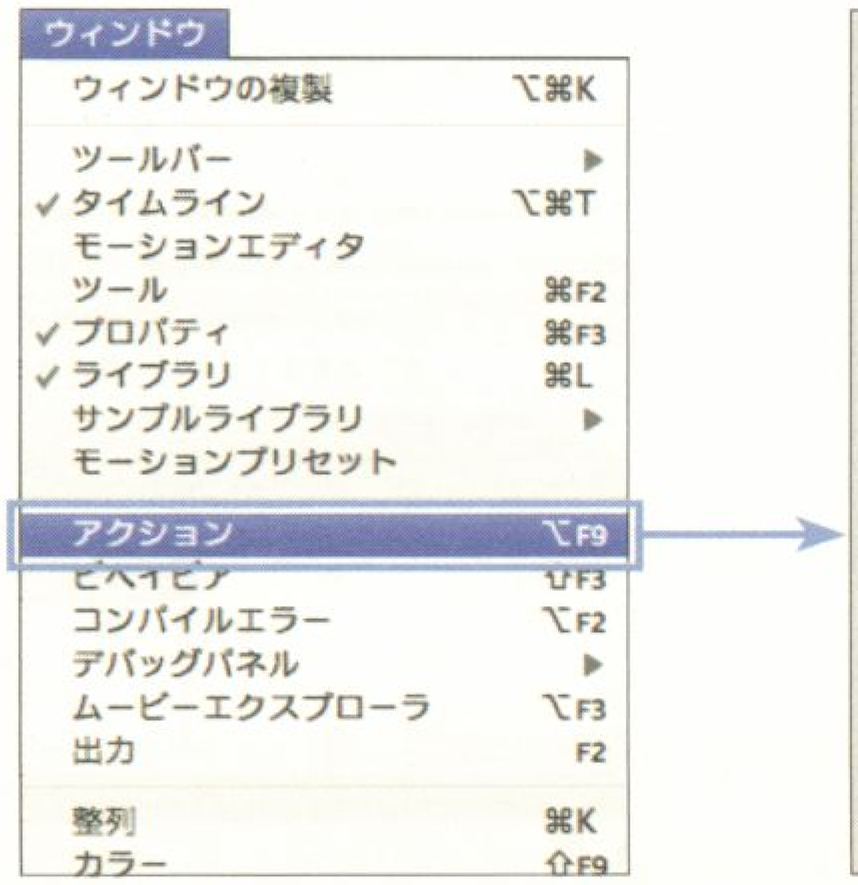


▶▶ アクションパネルの開き方

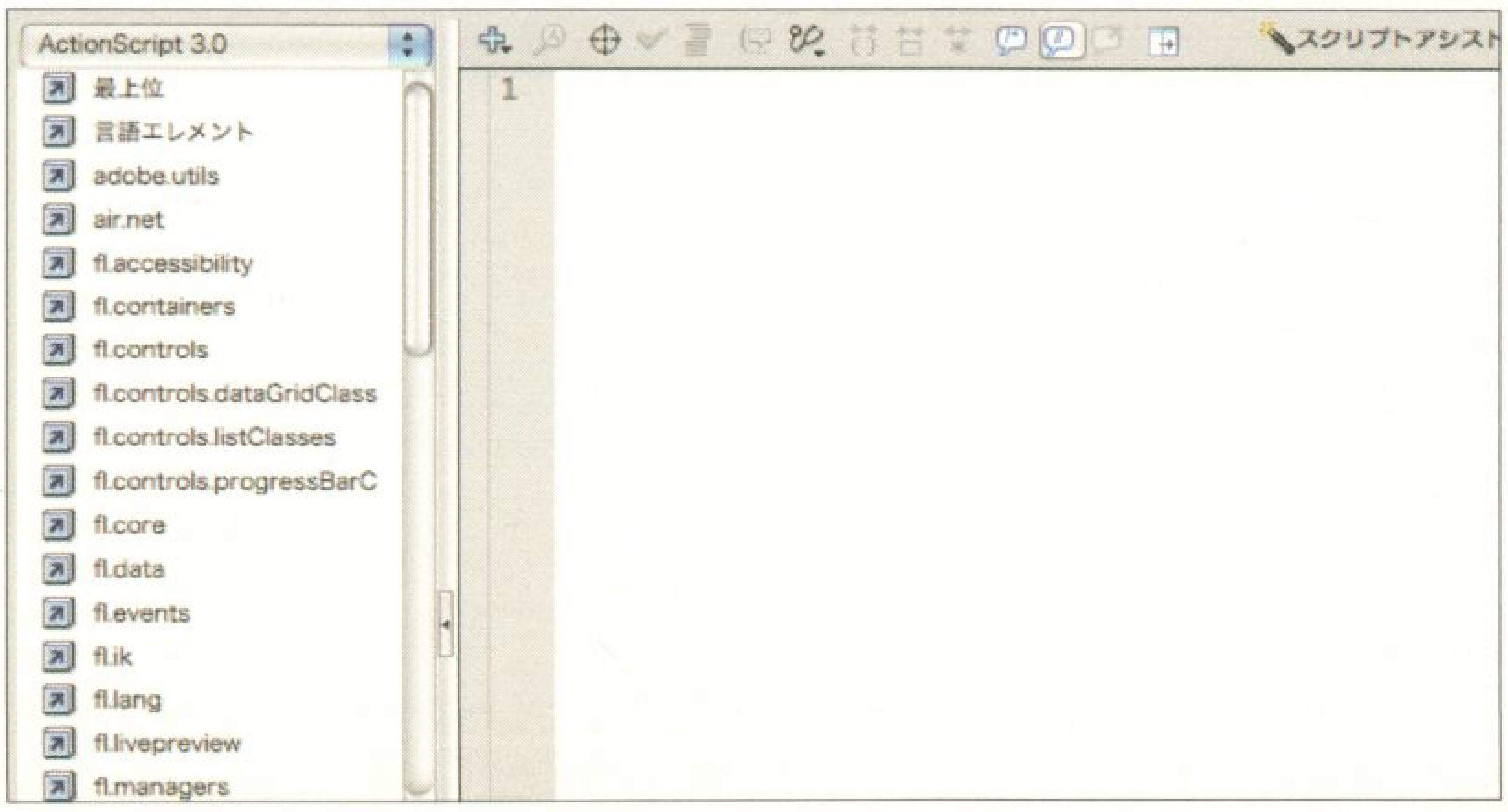
アクションパネルは、スクリプトを記述したいタイムラインを選択し、メニューの［ウィンドウ→アクション］を選択して開きます。

①キーフレームを選択する





②［ウィンドウ→アクション］を選択



③アクションパネルが開く

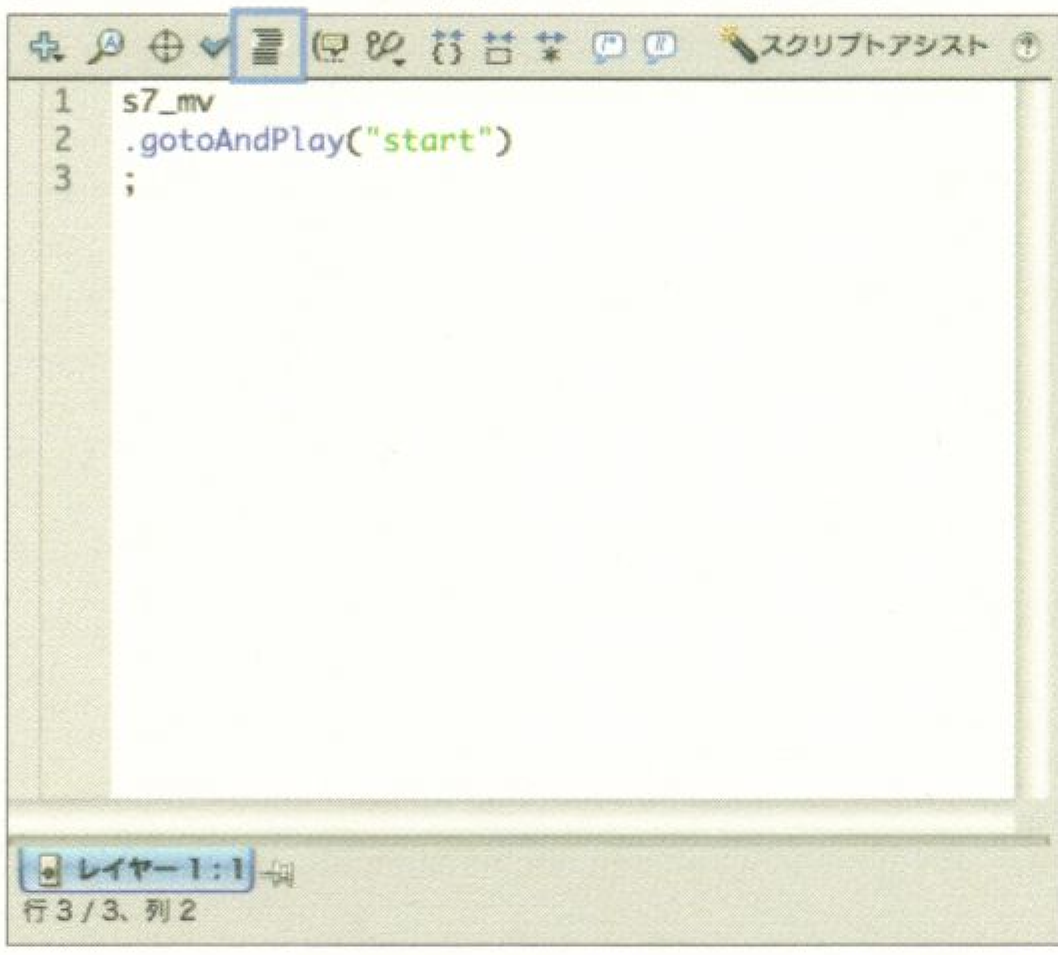
▶▶ アクションパネルの機能の使い方

ここでは最も使用頻度の高い「自動フォーマット」と「スクリプトアシスト」について、使い方を説明します。

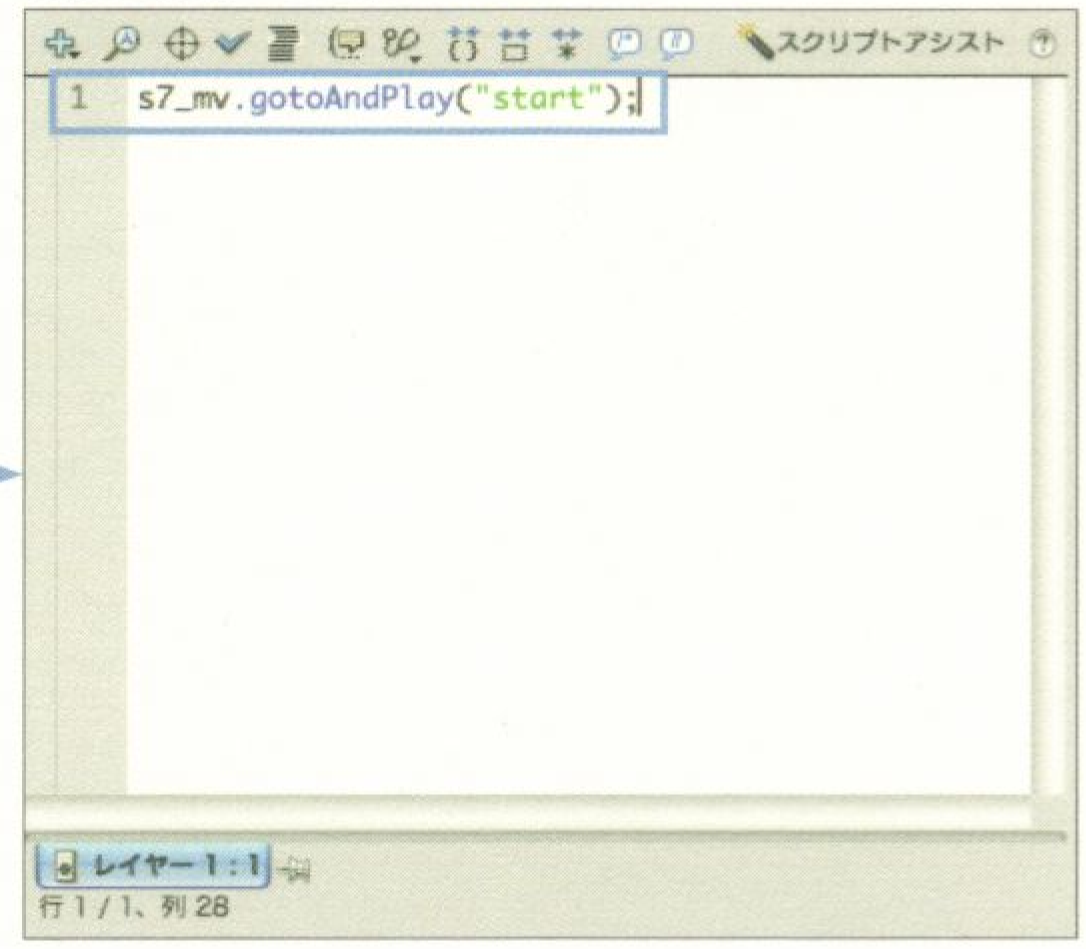
自動フォーマット

シンタックス「；」に合わせて、改行やインデントなどでスクリプトを整頓し、読みやすくさせる機能です。

①アクションパネルの「自動フォーマット」を選択



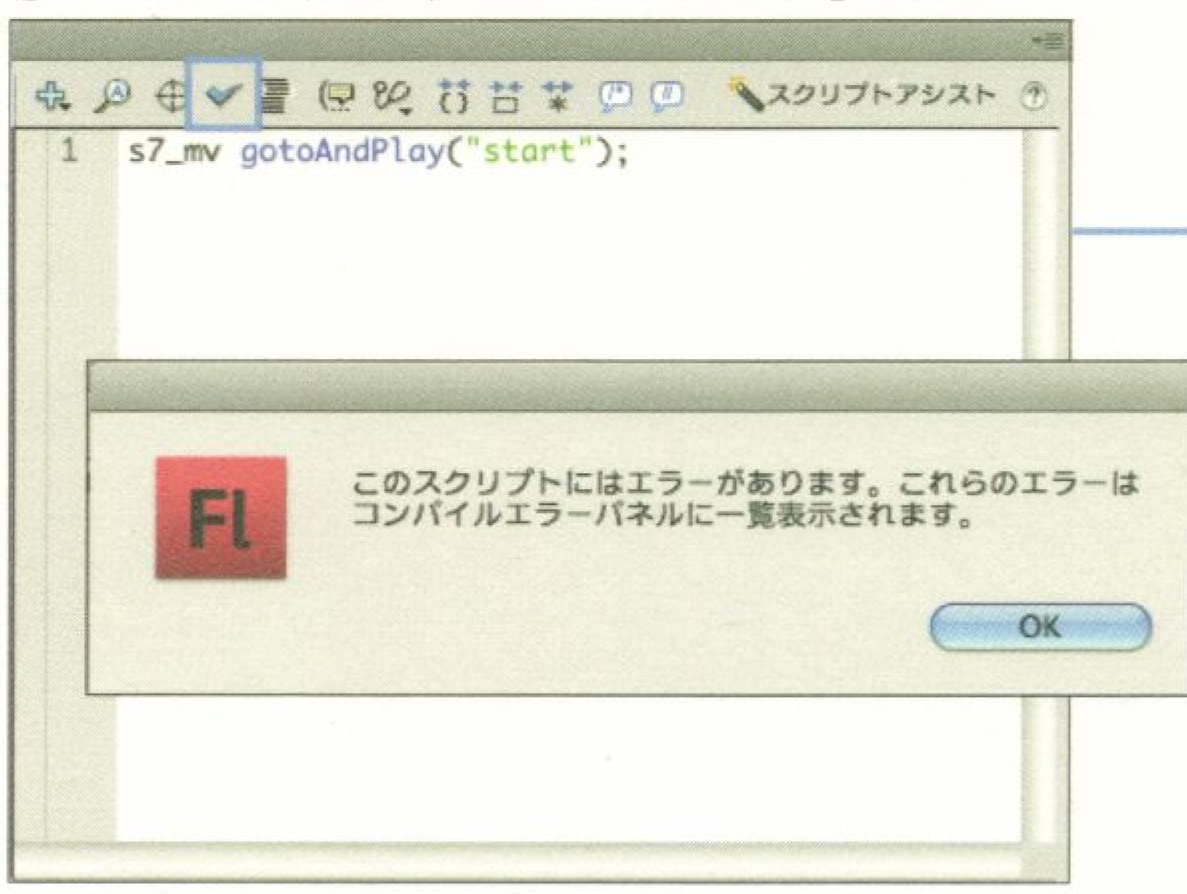
②記述内容が整頓される



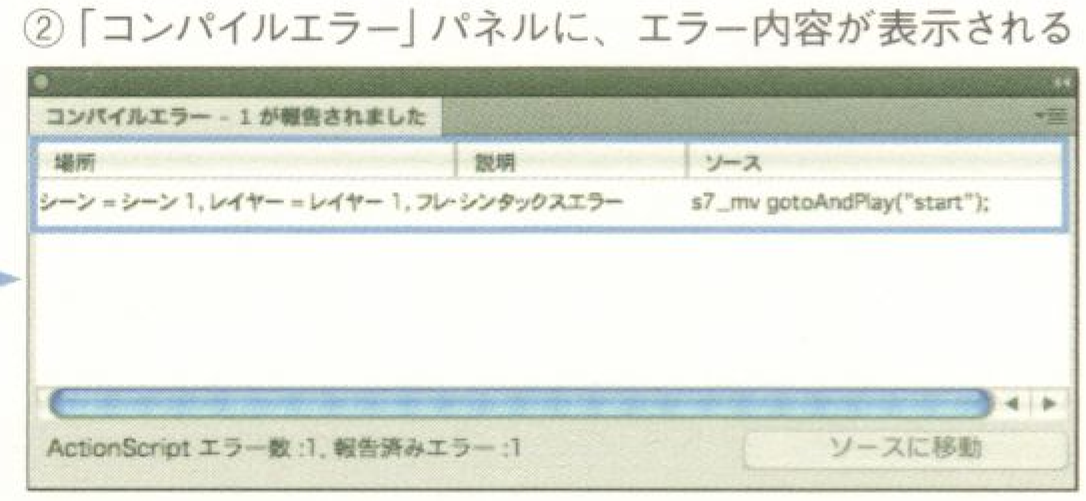
シンタックスチェック

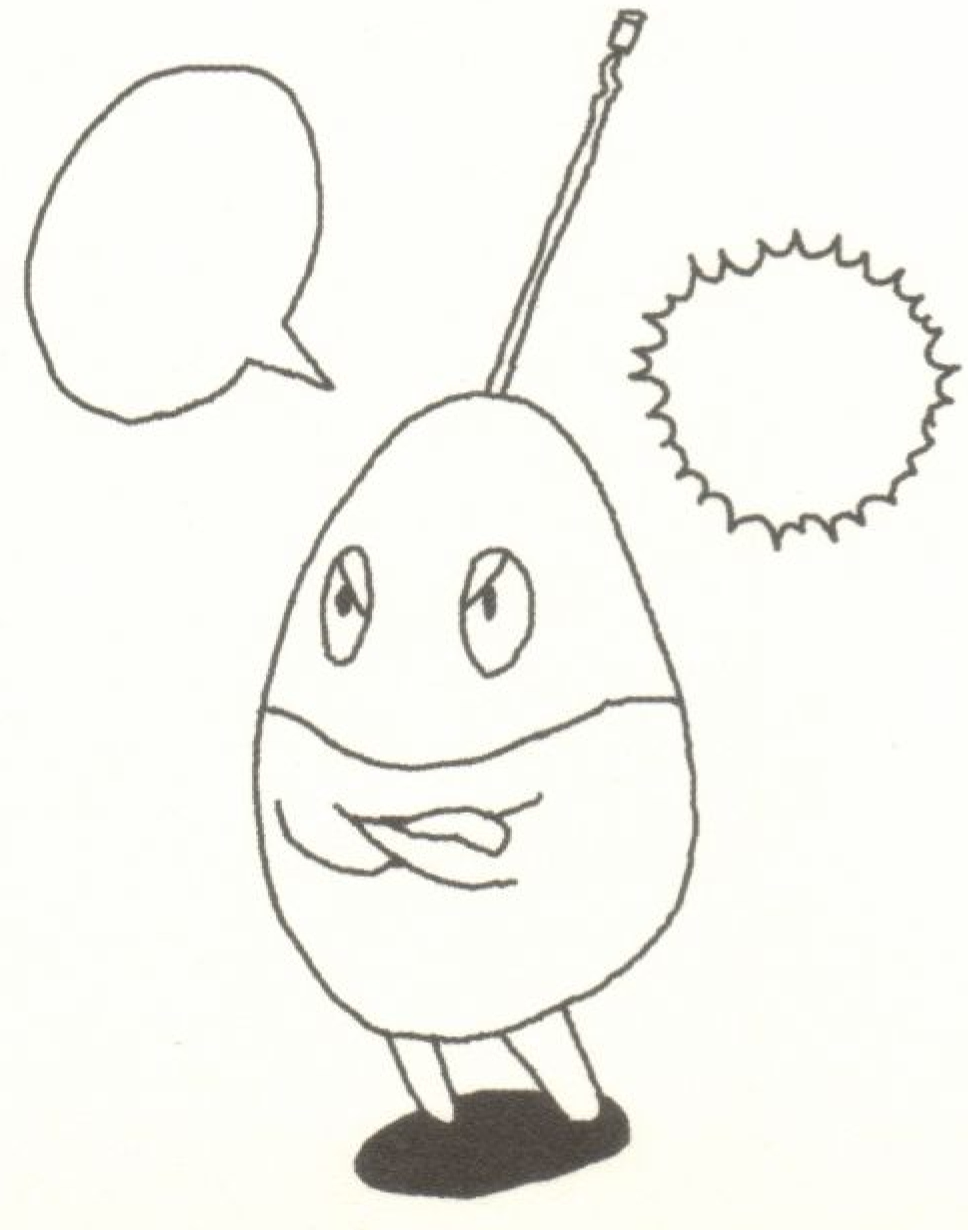
スクリプトにエラーがあった場合、エラーの場所と説明が報告されます。

①アクションパネルの「シンタックスチェック」を選択



②「コンパイルエラー」パネルに、エラー内容が表示される





ActionScriptの考え方

デバッグの方法 ── trace

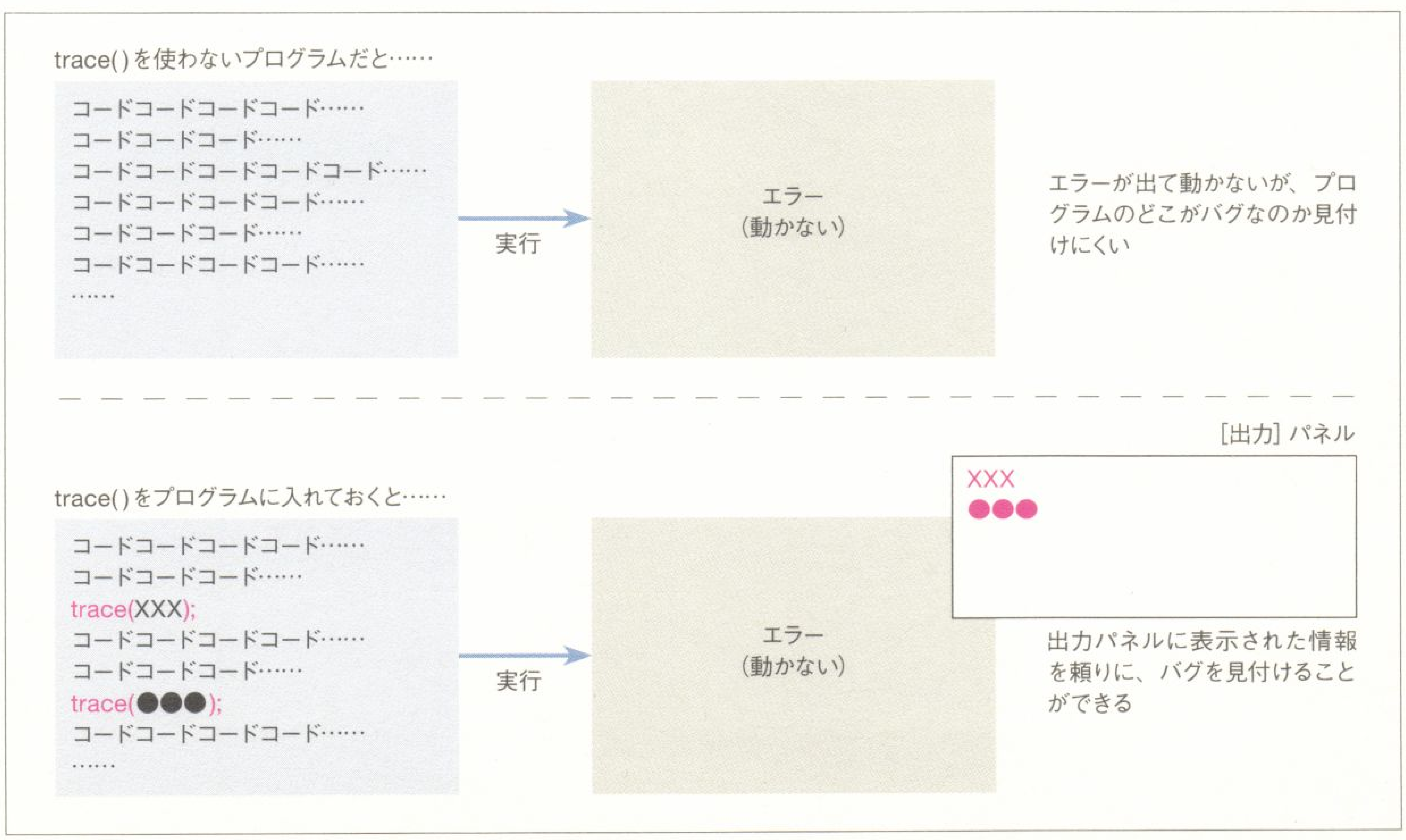
プログラミングは動いて当たり前、となれば良いのですが、どこかに間違いがあって、動かなくなることがあります。そのようなときに間違っている場所を見つけて修正することを「デバッグ」と言います。ここではtrace( )ステートメントの使い方と、それを使ったデバッグの方法を説明します。

合わせて読むと便利な項目

■ 変数 ……………………………… P094へ
■ 配列 ……………………………… P115へ
■ イベントリスナ…………………… P090へ
■ for ……………………………… P112へ
■ ENTER_FRAME …………………… P121へ

trace()を使ったデバッグの考え方

trace( )をプログラムに入れておくと、
Flashの［出力］パネルに表示される情報を使って、バグを見付けることができます。



▶▶ trace()ステートメントの使い方

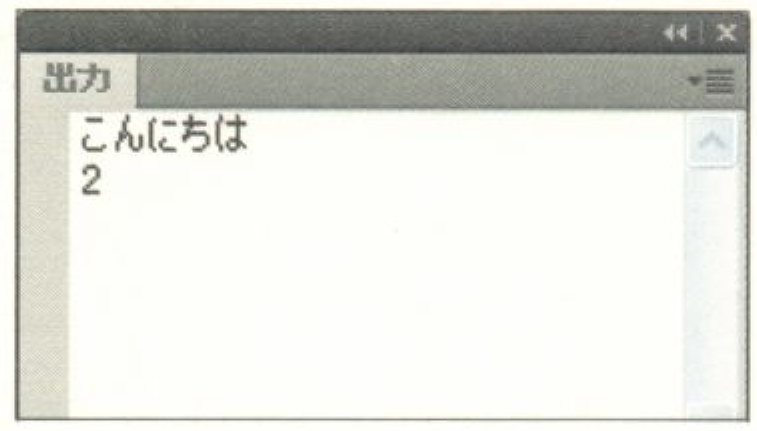
trace( )の使い方を見ていきましょう。

文字列や数値を表示させる

使用例

```
trace("こんにちは");
⬆出力パネルに「こんにちは」と表示させる（文字列は " " で囲む）
trace(2);
⬆ 数字を表示させる場合は " " は付けない
```

実行結果

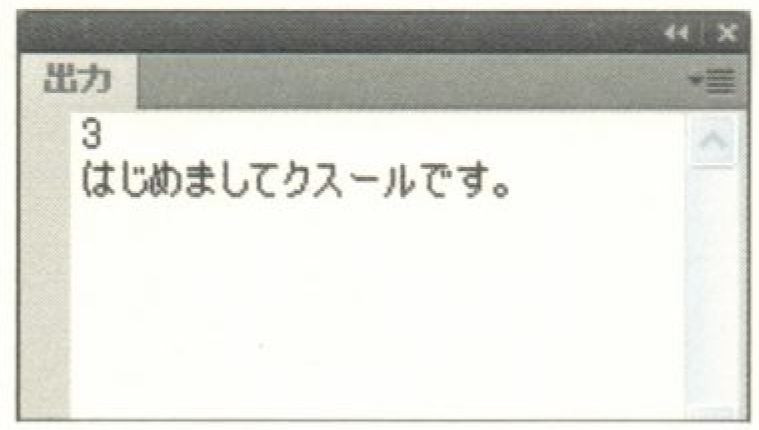


計算結果や連結させた文字列を配置する

使用例

```
trace(1+2);
⬆ 演算もできます
trace("はじめまして" + "クスールです。");
⬆ 文字列をつなげることもできる
```

実行結果



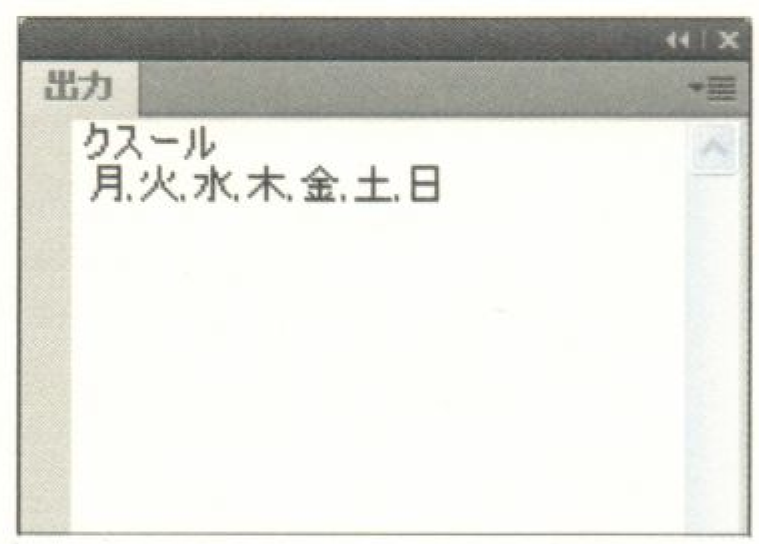
変数や配列を表示させる

使用例

```
var schoolName:String = "クスール"; ⬅変数schoolNameを作成
var myDay:Array = ["月", "火", "水","木","金","土","日"];
⬆ 配列myDayを作成

trace(schoolName); ⬅ 変数schoolNameの値を表示
trace(myDay); ⬅ 配列myDayの値を表示
```

実行結果

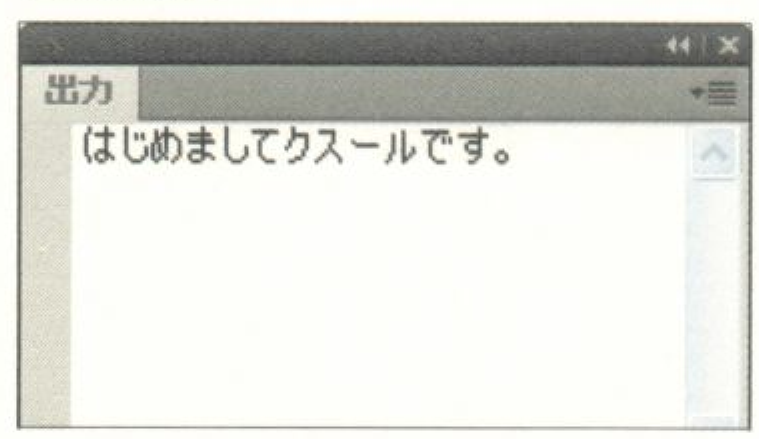


変数と文字列を連結させて表示させる

使用例

```
var schoolName:String = "クスール";
trace("はじめまして" + schoolName + "です。");
⬆ 文字列と変数をつなげることもできる
```

実行結果

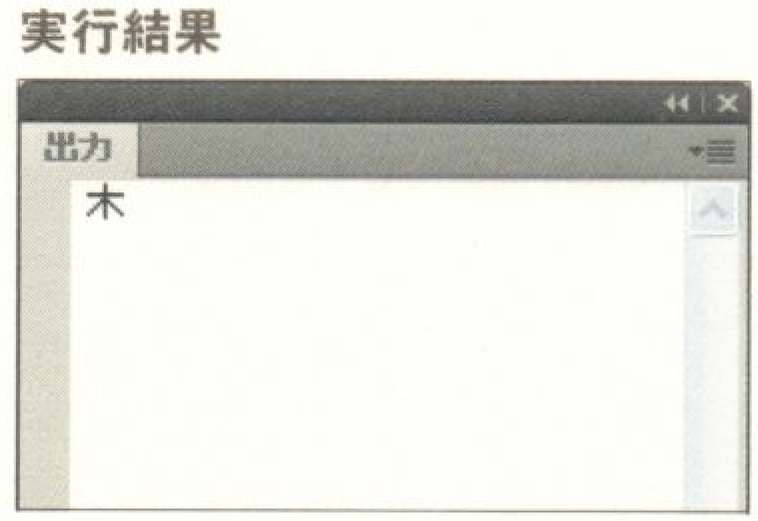


配列の中の特定の値を表示させる

使用例

```
var myDay:Array = ["月", "火", "水","木","金","土","日"];
⬆ 配列myDayを作成
trace(myDay[3]); ⬅ 配列myDayの4番目の値を表示させる
```

実行結果



trace()ステートメントを使ったデバッグの例

実際にtrace( )ステートメントを使ってデバッグをする例をいくつか紹介します。

▶▶ 関数が呼ばれているかを確認する

例えば、ボタンをクリックしたらムービークリップ内のタイムラインを再生する、というプログラムで、ボタンをクリックしても何も起こらない、という状況になったとします。この場合、まずは関数が正しく呼ばれているかどうかを確認します。

■イベントリスナ ………………………… P090へ

使用例

```
btn.addEventListener(MouseEvent.CLICK, startAnime);
function startAnime(event){
   trace("startAnimeが呼ばれました。");
   ↑ここが表示されれば、関数が呼ばれていることが分かるので、
     以下のスクリプトが間違っている可能性が高い

   (startAnimeの内容)
}
```

▶▶ インスタンスのプロパティを確認

スクリプトを使ってアニメーションをさせているムービークリップのx座標の値など、インスタンスのプロパティをtrace( )ステートメントを使って確認することができます。

■ENTER_FRAMEの使い方 …………… P121へ

使用例

```
addEventListener(Event.ENTER_FRAME, moveMc);

function moveMc(event){
   if(mc.x < stage.stageWidth){
      mc.x += 5;
      trace(mc.x);
      ↑連続移動しているmcのx座標を確認できる
   } else {
      removeEventListener(Event.ENTER_FRAME, moveMc);
   }
}
```

▶▶ for文の処理中に変数の値を確認

forを使った繰り返し処理の中で、X回目の処理で使われている変数の値を知りたい場合、このようにtrace( )ステートメントを使用することで確認できます。

■forの書き方 ……………………………… P112へ

使用例

```
var nameArray:Array = ["佐藤","鈴木","田中","高橋"];

for(var i = 0; i < nameArray.length; i++){
   trace("処理" + i + ":   " + nameArray[i]);
   ↑X回目の処理と配列の値を確認できる
}
```

trace( )ステートメントを使ったファイルをパブリッシュした後、できたswfファイルをFlash Playerで再生しても、[出力]パネルは表示されません。とても便利な機能ですが、開発が終わり、swfをWebサイトで公開する際には、スクリプト内のtrace( )ステートメントは削除することをおススメします。
理由は、削除した方が再生のパフォーマンス(全体の処理速度)が若干向上すること、また最近のWebブラウザのプラグインなどで、swfファイル内のtrace( )ステートメントを[出力]パネルのように表示させることができてしまうことです。trace( )ステートメントを一括で削除する場合は、[パブリッシュ設定]を開き「Flash」タブにある「Trace アクションを省略」を選択します。

LECTURE 3-5

ActionScriptの考え方

クラス

ActionScript 3.0には、「クラス」と呼ばれるさまざまな機能や情報を持った仕組みがあらかじめ用意されており、これらを組み合わせることでいろいろなプログラムを実現することが可能となっています。

クラスが使われているサンプル

■ PRACTICE 5：
サムネイルを動的に配置する ……… P224へ

合わせて読むと便利な項目

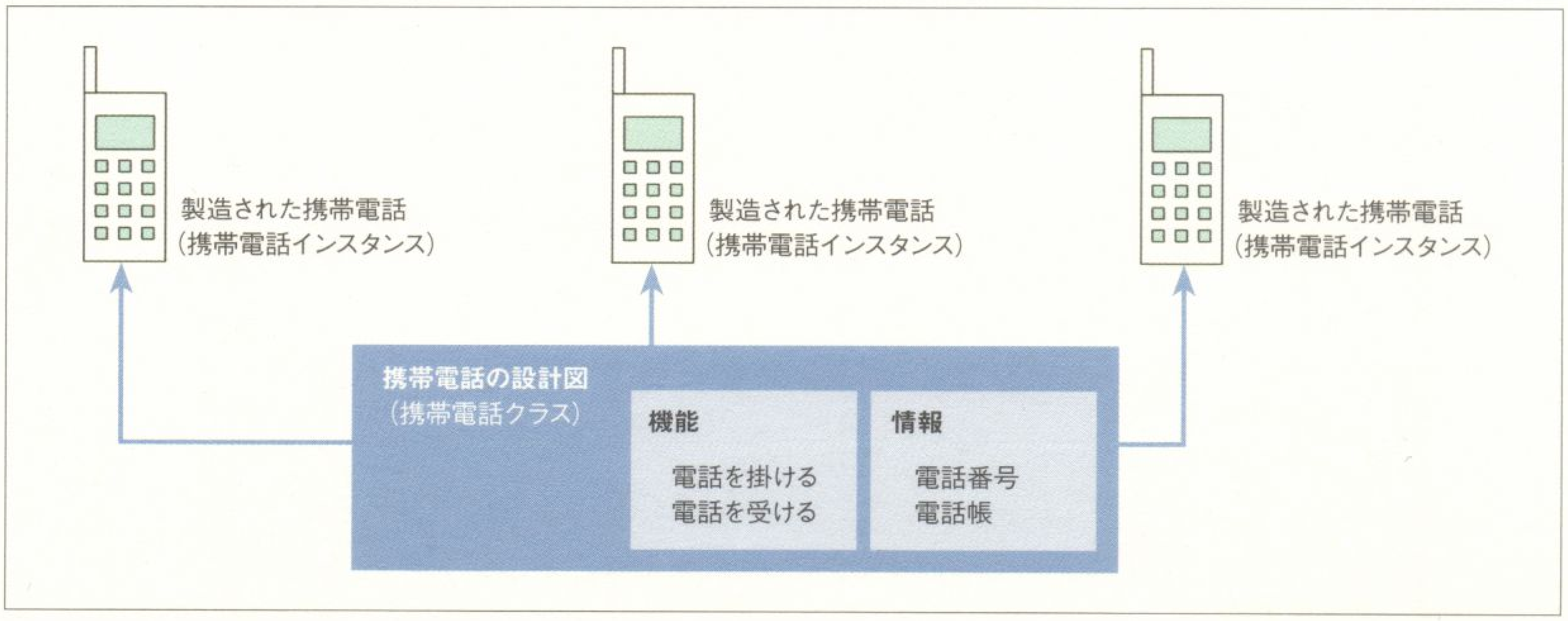
■ ActionScriptとは ………………… P064へ
■ 変数 ……………………………………… P094へ

クラスとは

「クラス」とは、プログラム上で使えた機能や情報をまとめて、あらかじめ作成された設計図のようなものです。この設計図をもとにさまざまな仕組みを作成することができます。

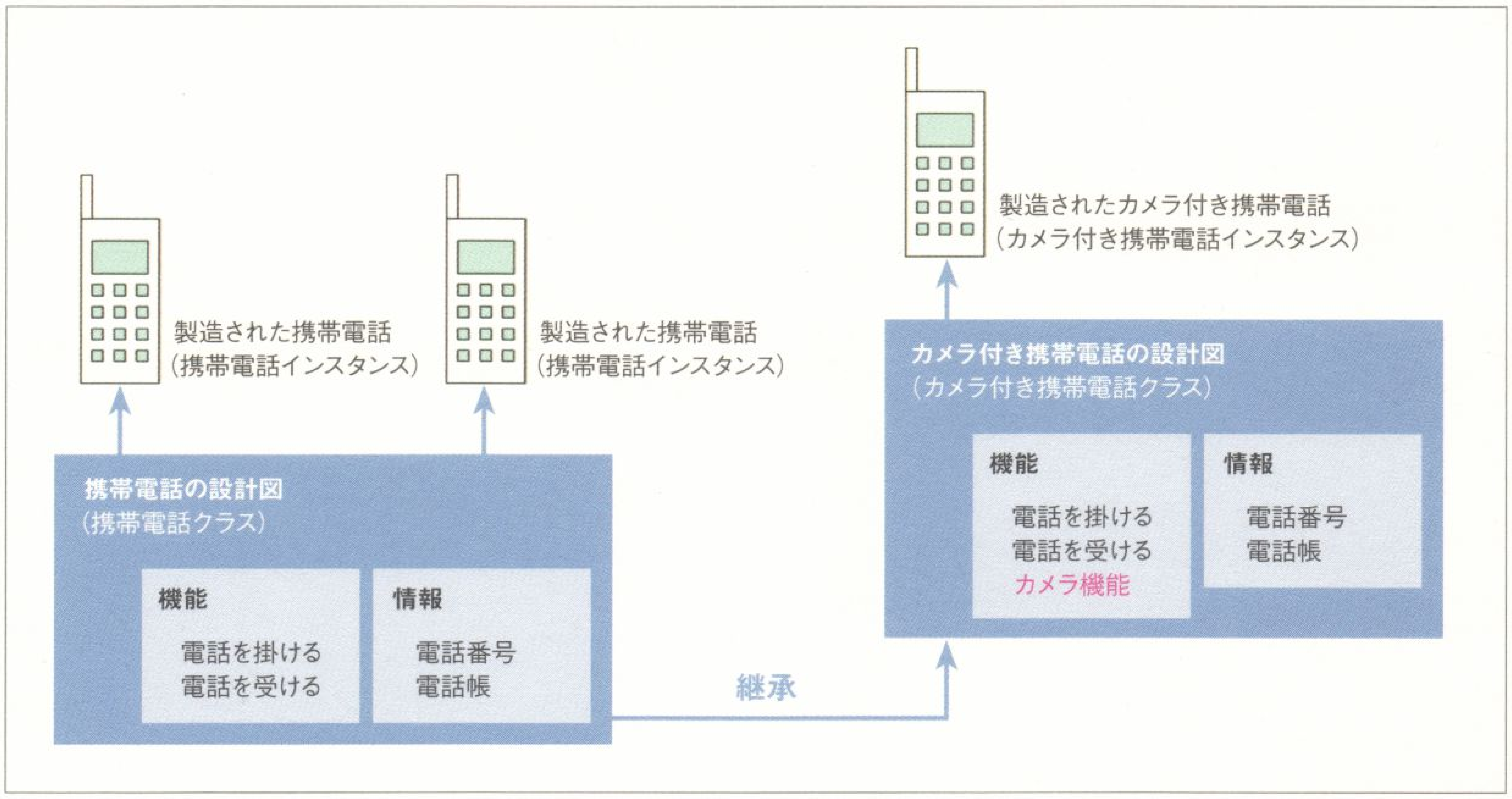
▶▶ クラスとインスタンス

実際の世界における、「携帯電話」を例にクラスの構造を考えてみます。
携帯電話には最低限必要な機能として「電話を掛ける」「電話を受ける」という機能が必要となります。また、「電話番号」や「電話帳」といった情報も必要です。これらの必要な機能や必要な情報を定めておいたものを「クラス」と呼びます。この「携帯電話クラス」を用いて、実際に使う携帯電話を作成していきます。この「携帯電話クラス」を実際に製品にすることで、実際に使える機能を持った携帯電話を作成することができます。このように「クラス」をもとに作成されたものを「インスタンス」と呼びます。



▶▶ 継承

「クラス」は、その機能に新たな機能、情報を追加して更に新しいクラスを作成することが可能です。
例えば、「携帯電話クラス」にカメラ撮影の機能を付けたいとなった場合は、「携帯電話クラス」に機能を追加して「カメラ付き携帯電話クラス」を作成することができます。機能や情報を加えた新しいクラスを作る際に、あるクラスをもとにすることを「継承」と呼びます。



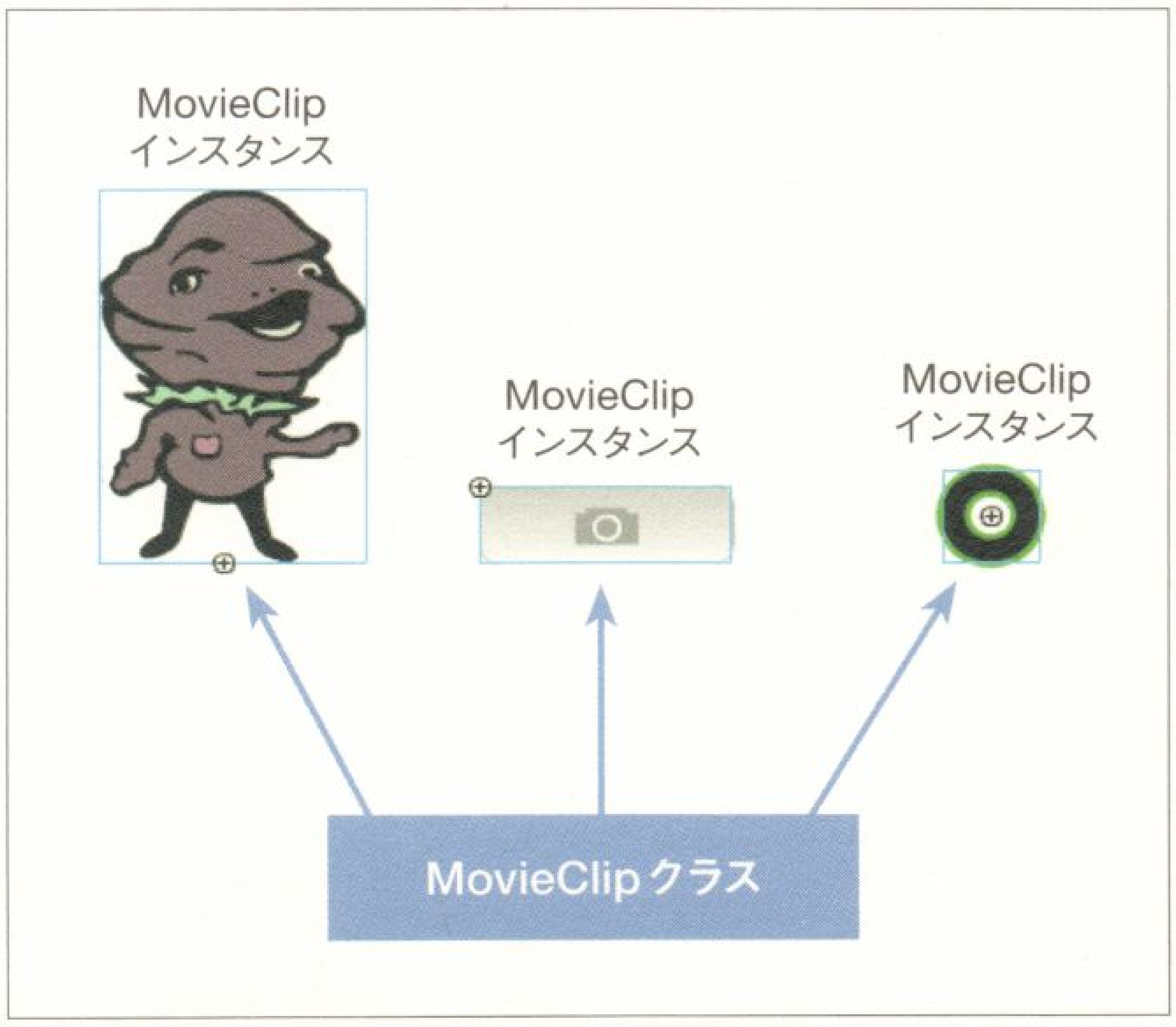
Flashにおけるクラス

Flashでのプログラミングでは、あらかじめ用意された「ビルトインクラス」や、自分や他人が新たに作成した「カスタムクラス」を組み合わせることでさまざまなコンテンツを作成することができます。

▶▶ MovieClipクラス

実際にActionScript 3.0で使われているクラスについて考えます。Flash上で画面に配置したムービークリップはすべてMovieClipクラスのインスタンスとなっています。

MovieClipクラスは「タイムラインの操作」や「ムービークリップ内の表示要素の制御」機能、「位置」や「大きさ」、「角度」などの情報を持っています。これにより画面上に配置されたすべてのムービークリップで同じように「タイムラインの操作」や「表示要素の制御」、「位置」や「大きさ」などの情報の取得が可能となります。



ビルトインクラスとカスタムクラス

ActionScript 3.0のクラスは大きく2種類に分けることができます。「ビルトインクラス」と「カスタムクラス(ユーザー定義クラス)」です。
「ビルトインクラス」はActionScriptにあらかじめ作られたクラスのことを指し、これらのクラスを用いて、さまざまなプログラミングを行うことができます。
また「カスタムクラス」はユーザーによって新たに作成されたクラスを指します。自分が作成したクラスだけでなく、ActionScriptライブラリとして公開されている「Progression」や「PaperVision3D」などに含まれるクラスも「カスタムクラス」と呼ばれます。

本書で登場する主なビルトインクラスとその使い方

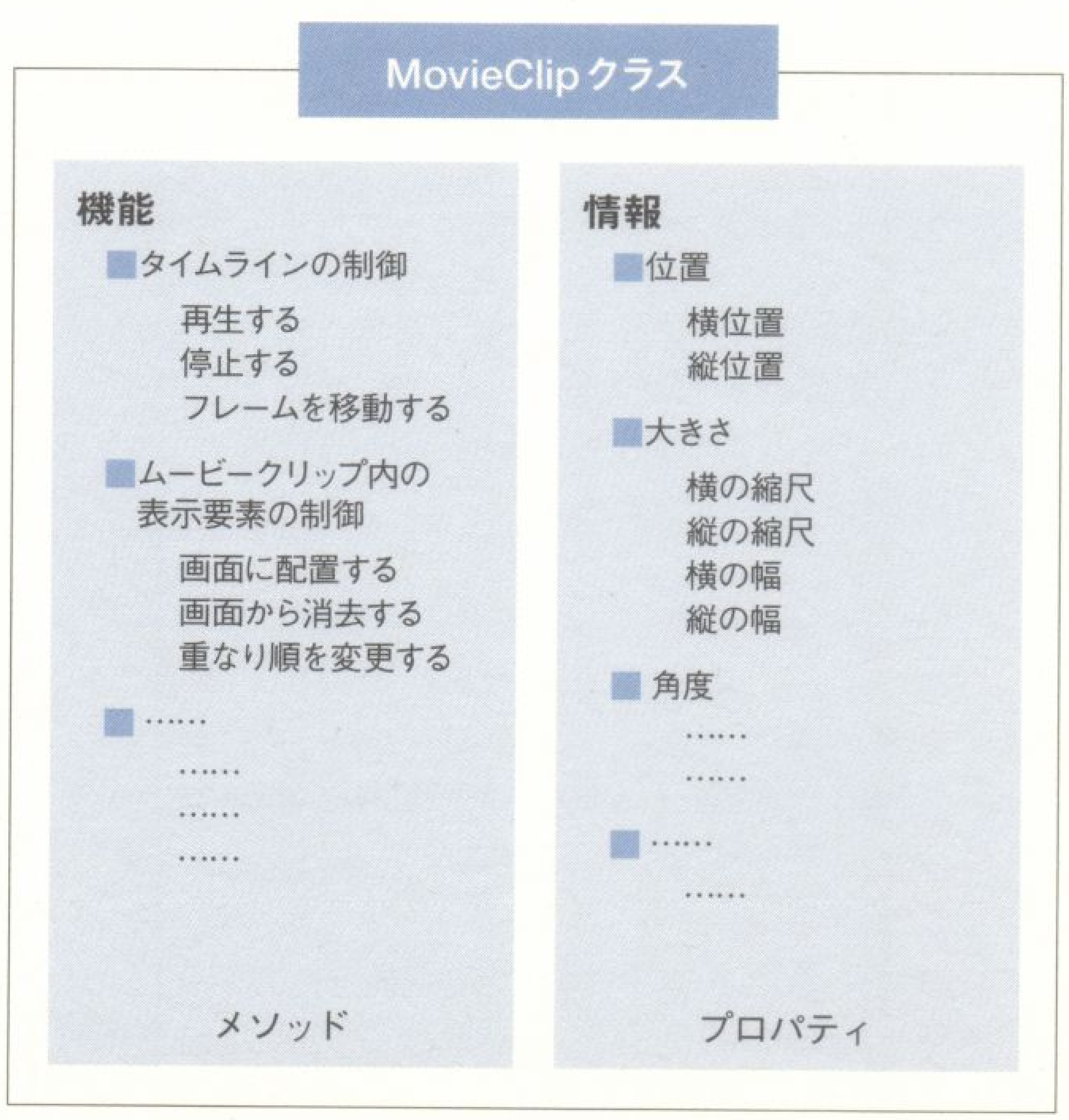
クラス名		主な使い方	
MovieClip	ムービークリップ	MovieClip.play( )	⇐ ムービークリップを再生する
Array	配列	var array = new Array( )	⇐ 空の配列を作成
Object	オブジェクト	var object = new Object( )	⇐ 空のオブジェクトを作成
TextField	テキストフィールド	TextField.text = "テキスト"	⇐ テキストフィールドの表示テキストを文字列「テキスト」に設定
Stage	ステージ	Stage.stageWidth	⇐ ステージの横幅
Camera	カメラ	Camera.getCamera( )	⇐ 使用可能なカメラを取得する
Date	日付	Date.getDate( )	⇐ 日付を取得する

プロパティとメソッド

それぞれのクラスは「プロパティ」と呼ばれる情報、「メソッド」と呼ばれる機能を持っています。
これらを実行したり、取得・設定することでさまざまなプログラミングが実現できます。

各クラスが持っている機能のことを「メソッド」と呼びます。例えばMovieClipクラスは、タイムラインを制御するメソッドとして「play( )」「gotoAndStop( )」などを、表示要素を制御するメソッドとして「addChild( )」「removeChild( )」「setChildIndex( )」などを持っています。
また、各クラスが持っている情報は「プロパティ」と呼ばれます。同じくMovieClipクラスの例では、位置情報のプロパティ「x」「y」、大きさのプロパティ「scaleX」「scaleY」などを持っています。

■ MovieClipのプロパティ ………………… P084へ

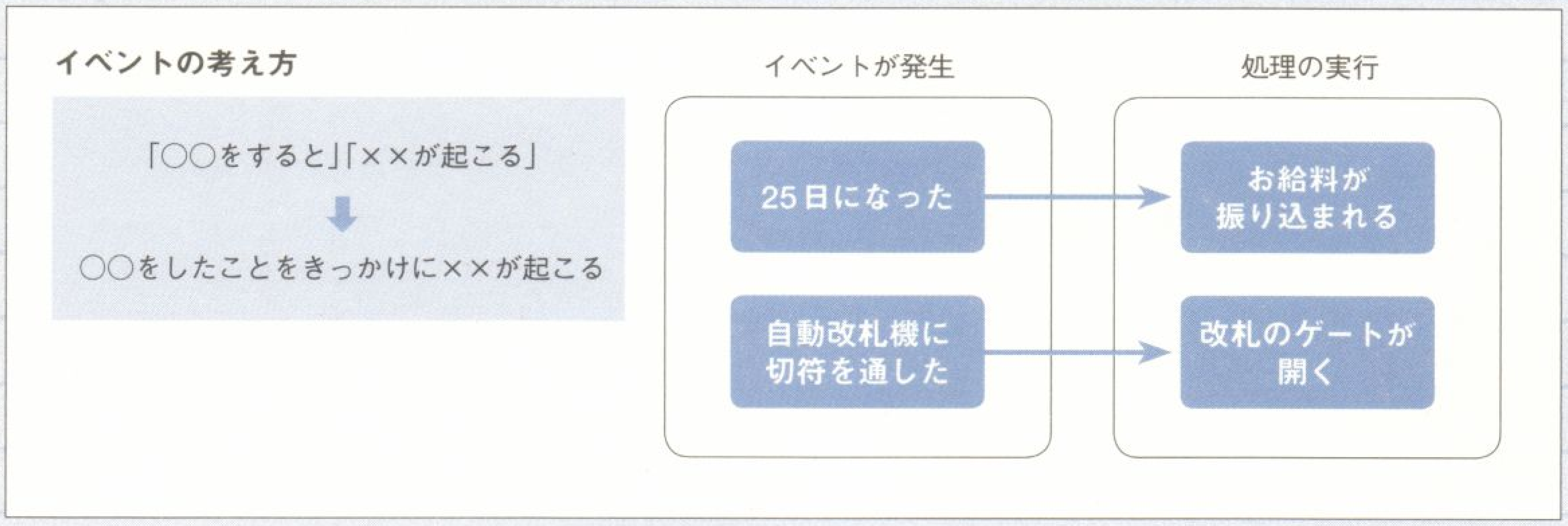


LECTURE 3-6

ActionScriptの考え方

イベントの概念

ActionScript 3.0を学ぶ上で、イベント処理の考え方を理解することは非常に重要です。とはいえ、あまり難しく考えずに、概要から理解していきましょう。まずは、身の回りの出来事で、イベント処理の考え方を見てみましょう。イベント処理の考え方を簡単にまとめると、「何かが起こったことをきっかけに、何かを実行する」ということになります。このように考えると、普段の生活の中でもイベント処理は発生しています。例えば、「毎月25日にはお給料が振り込まれる」、「自動改札機に切符を入れるとゲートが開く」etc..... 。このように「誰かが」「○○をすると」「××が起こる」という一連の流れのなかで、「○○をする」ことをきっかけに「××が起こる」ということがイベント処理の大きな流れになります。



イベントが使われているサンプル

■ PRACTICE 2：
ボタンの動きをスクリプトで制御する… P196へ
■ PRACTICE 6：
アニメーションを組み立てる ………… P241へ

合わせて読むと便利な項目

■ イベントリスナ……………………… P090へ

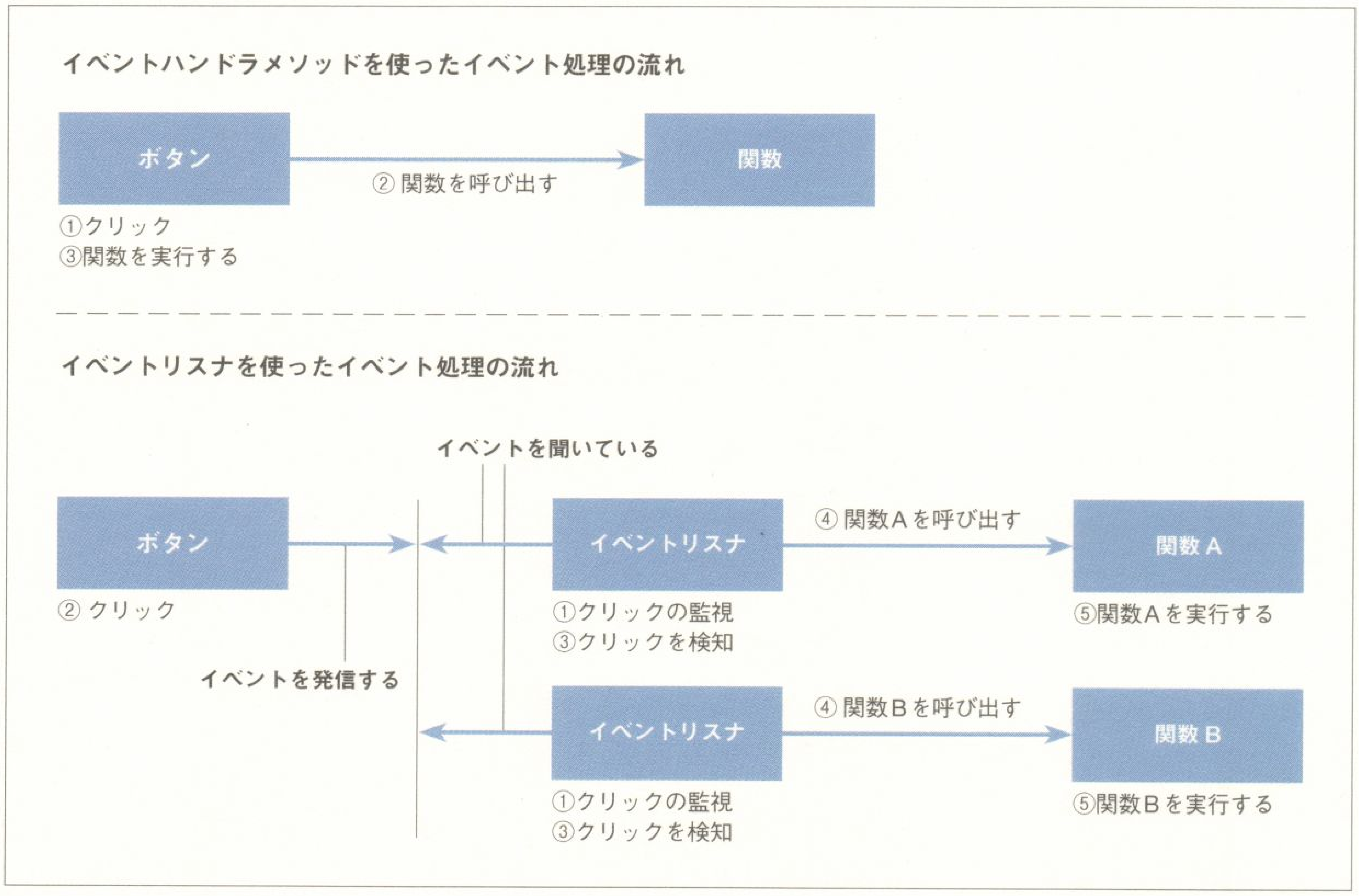
イベントとは

Flashでのイベント処理の流れを図を使って見ていきます。

▶▶ イベントハンドラとイベントリスナ

例えば「ボタンを押したら次の画面に移動する」とか、「画像ファイルの読み込みが完了したら読み込んだ画像を開始する」というように、「何かが起こると、それをきっかけにして実行する」という処理が数多くあります。
ActionScript 3.0でこのようなイベント処理を行う際には、イベントリスナという仕組みを使います。リスナーと言う名前から、「イベントの発生を常に聞いていて、イベントが発生したらそれを関数に伝える仕組み」をイメージすると、分かりやすいと思います。
ActionScript 2.0ではイベントハンドラメソッドという仕組みを利用してイベント処理を行っていましたが、ActionScript 3.0ではイベントリスナを使ったイベント処理に統一されました。「ボタンを押したら○○する」という処理の流れを例に、2つの違いを下の図にまとめます。

イベントハンドラメソッドを使ったイベント処理の場合、ボタンが関数を呼びに行くところまでを担当しますが、イベントリスナを使った処理の場合はボタンはただ「クリックされた」という情報を発信すれば良く、イベントリスナがその情報をキャッチして関数を呼びに行く、という違いがあります。
イベントリスナを使った処理は、複数の処理を同時に行うことができ、かつそれぞれの処理の流れを個別に消したりすることができます。



Flashでのイベントの使い方

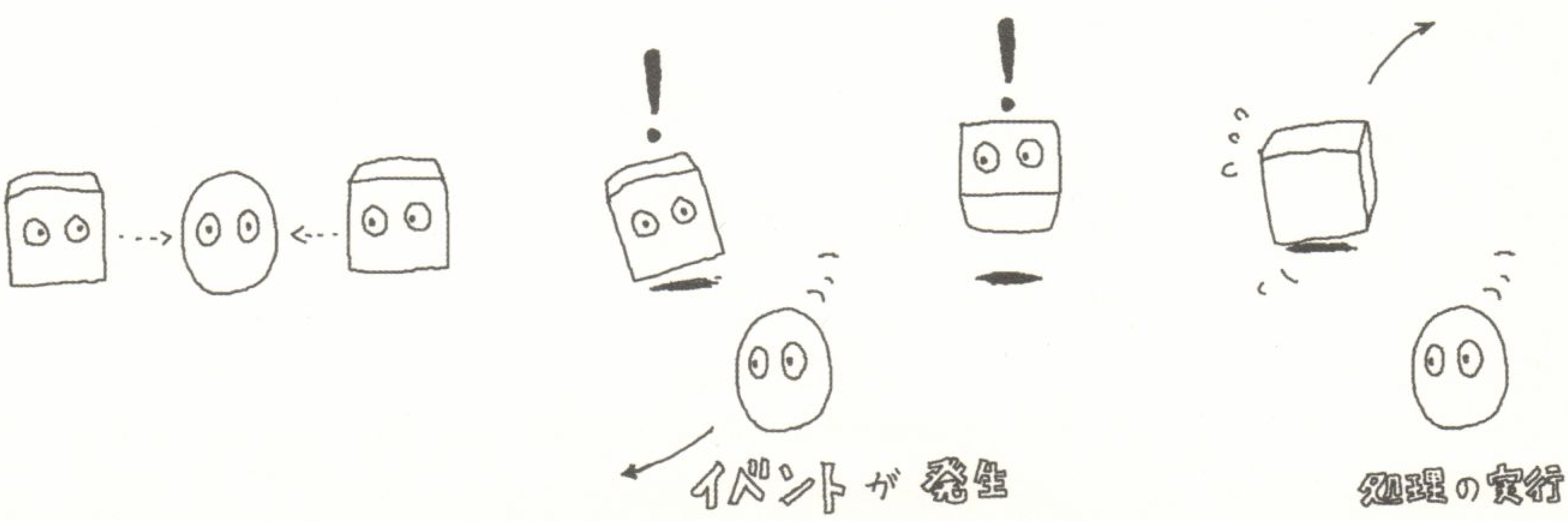
次に、「ボタンをクリックしたら関数eventTest( )を実行する」というイベント処理の流れを、ActionScriptのイベント処理の用語に置き換えて考えてみます。
イベントリスナを使ったイベント処理に必要な要素は、イベントがどこで発生するかを示す「ターゲット」、クリック、キー操作、ファイル読み込みなどの「イベントの種類」、そしてイベントが発生した際に実行される「リスナー関数」の3種類です。
プログラムの詳細については、「イベントリスナ」の項で解説しますので、そちらを参照してください。

■ イベントリスナ……………………………… P090へ

イベント処理に必要な要素

- ・イベントのターゲットインスタンス（誰が）
- ・イベントの種類（○○したら）
- ・リスナー関数（××する）





LECTURE 4-1

プログラミング［基本］

タイムラインを制御する

ActionScriptを使ってタイムラインを制御する方法を解説します。タイムラインはムービーを再生すると、1フレームから順にコマ送りで再生し、最終フレームまで再生するとまた1フレーム目に戻り（ループ）ますが、この再生・停止をActionScriptを使って制御することができます。また、特定のフレームから再生を始めたり、特定のフレームに飛んでそこで停止、ということもできます。

タイムライン制御が使われているサンプル

■ PRACTICE 2：メインイメージの動きをスクリプトで制御する ………………… P197へ

合わせて読むと便利な項目

■ アニメーションを作る ………………… P046へ

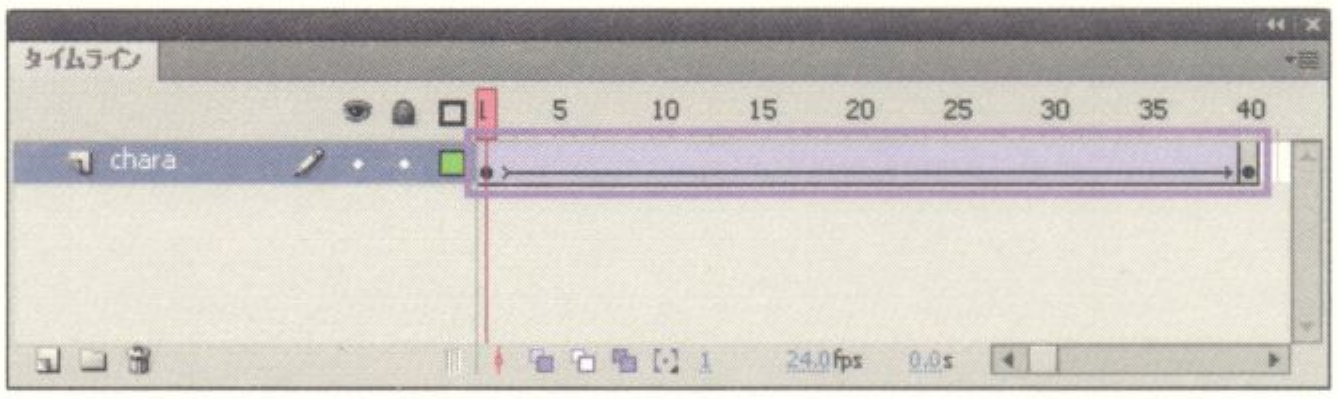
■ MovieClipを制御する ………………… P083へ

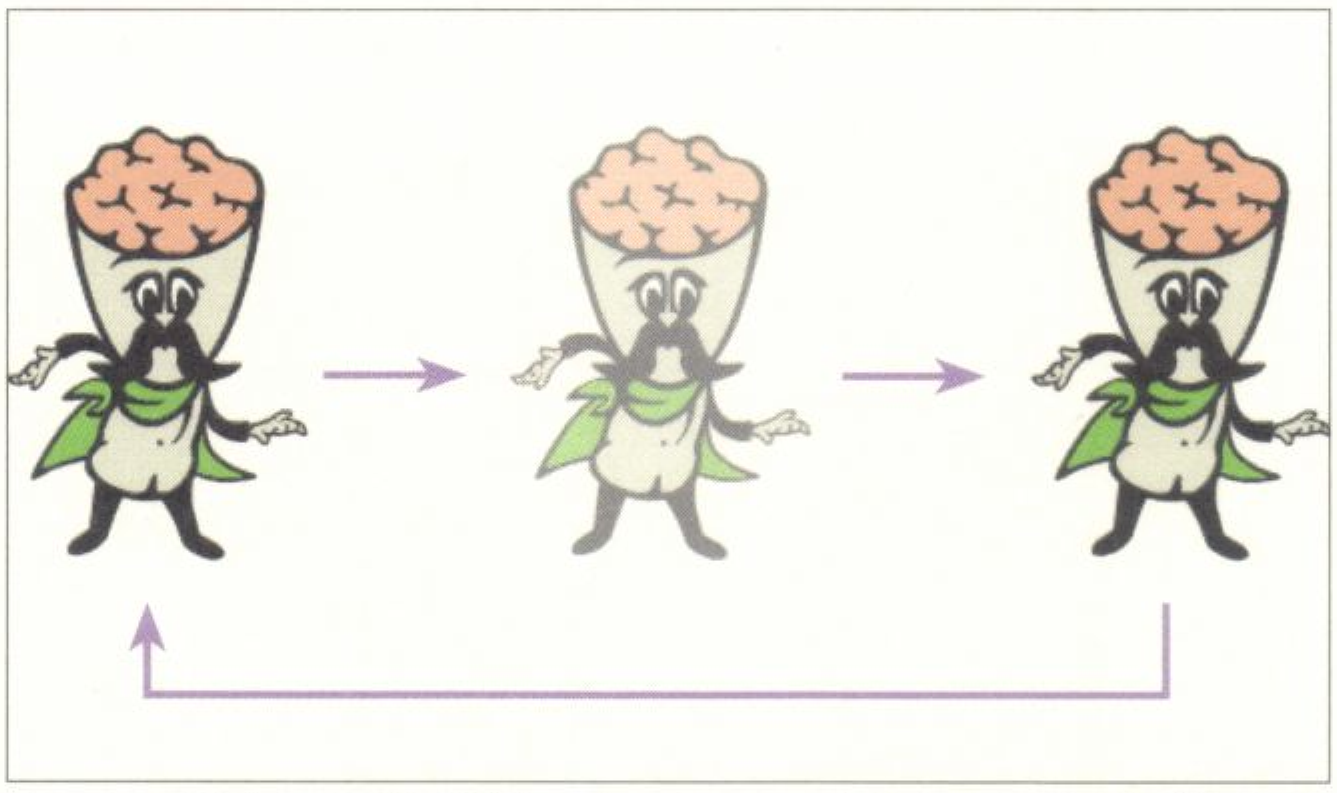
タイムラインを制御するActionScript

ActionScriptを使うことで、ムービーを再生/停止したり、特定のフレームにジャンプしたりできます。主要なものを紹介しましょう。

▶▶ play( )、stop( )

タイムラインの再生を開始するにはplay( )、再生ヘッドを停止させるにはstop( )を使います。
まずはキャラクターが横に移動するアニメーションをクラシックトゥイーンを使って作ります。サンプルでは、40フレームで左から右へ移動するアニメーションを作りました。
この状態でムービープレビューすると、40フレーム目の次はまた1フレーム目から再生が始まり、アニメーションがループします。

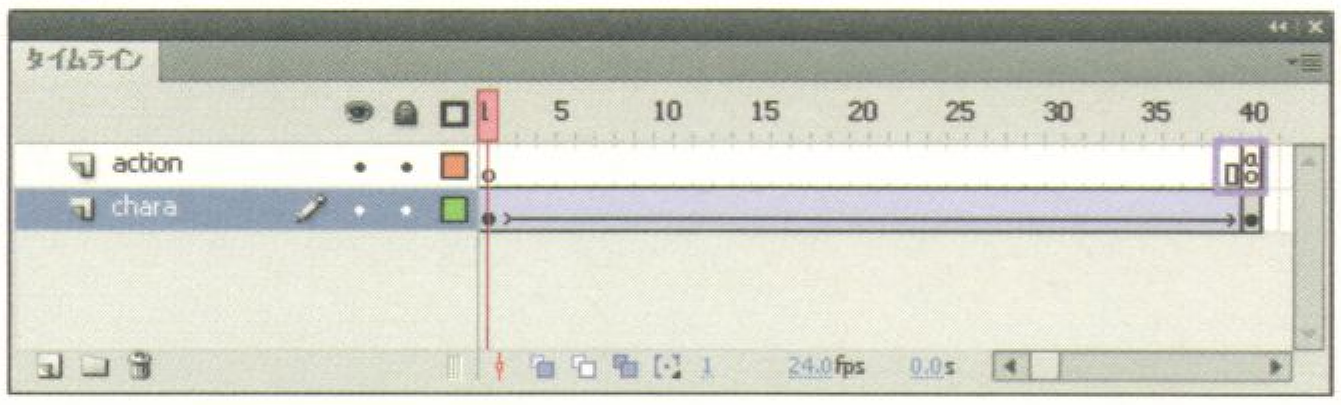


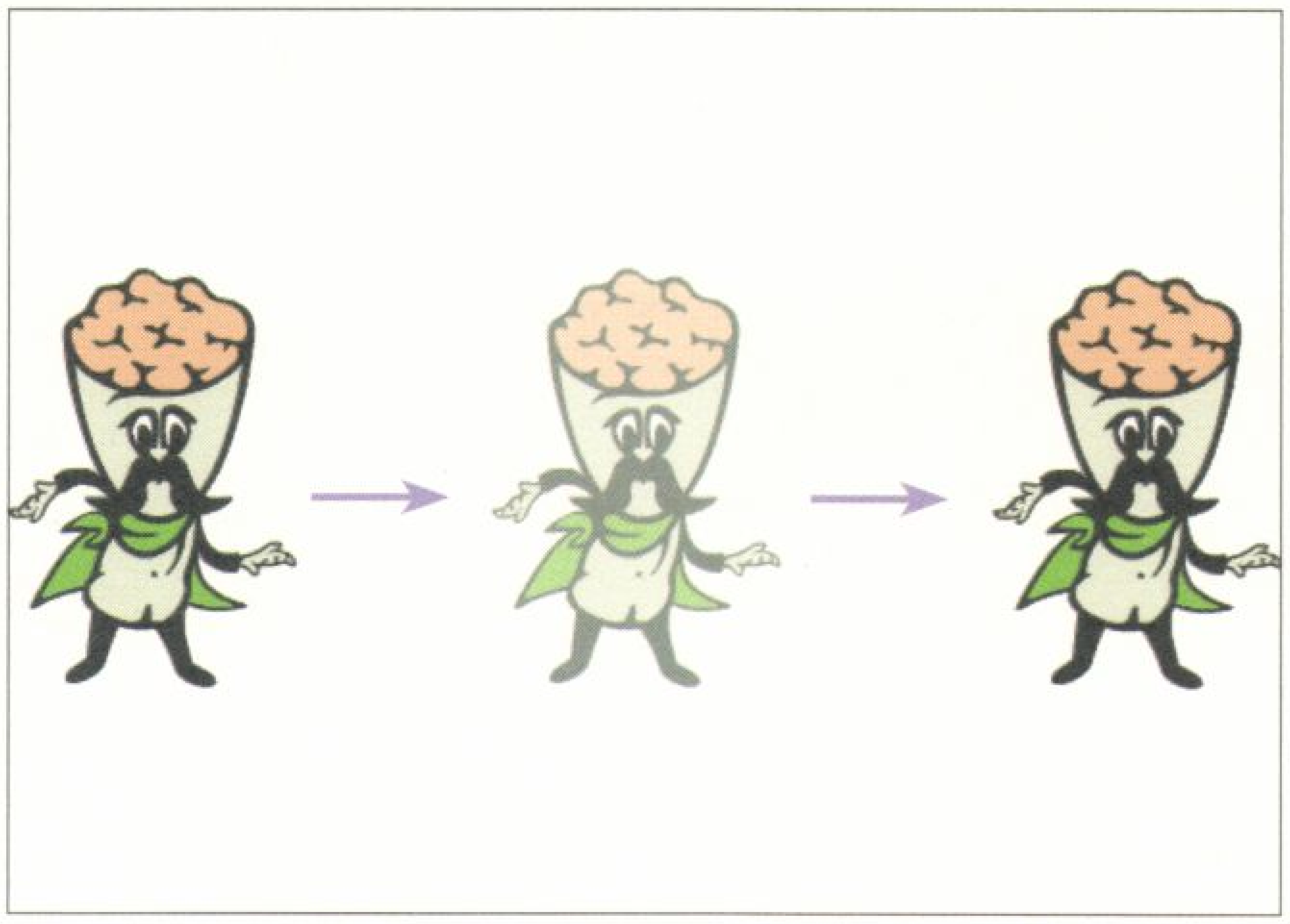


40フレーム目まで再生が終わると、1フレーム目からまた再生が始まる

ここで、最後の40フレーム目に、stop( )を記述してムービープレビューをすると、キャラクターが右に行った時点でアニメーションの再生が停止され、ループが止まります。

```
stop(); ⬅ 現在のフレームで再生を停止させる
```





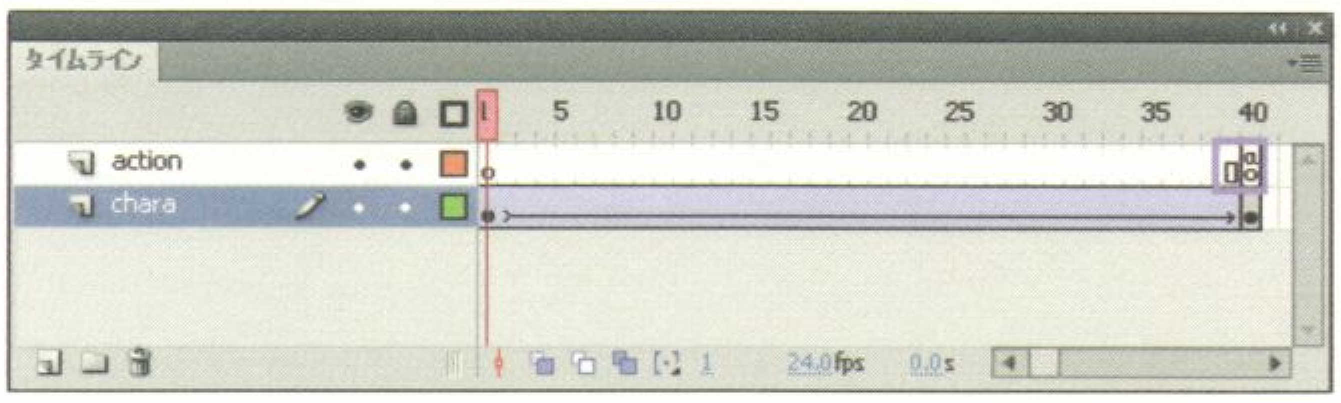
40フレーム目で再生ヘッドが停止し、アニメーションが止まる

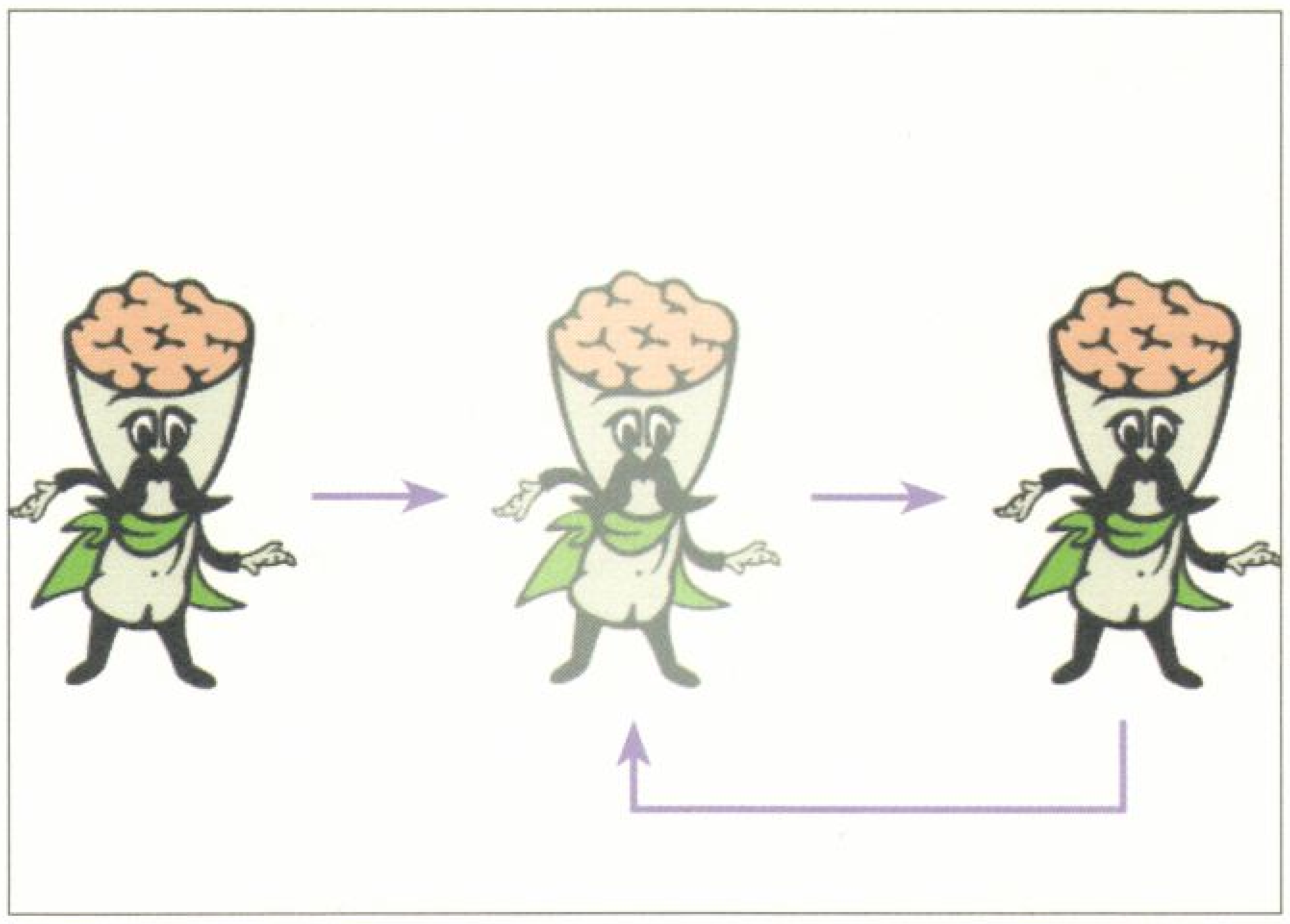
▶▶ gotoAndPlay( ) ―― (1)

特定のフレームに再生ヘッドをジャンプさせ、そこから再生を始めるには、gotoAndPlay( )を使います。

stop( )の項で作成したサンプルで、40フレーム目に記述したスクリプトを、gotoAndPlay(20)と書き換えると、右まで行ったキャラクターが真中(20フレーム目の位置)に現れ、そこから再生が始まり、アニメーションがループします。

```
gotoAndPlay(20);
⬆ ( )内で指定したフレームに再生ヘッドがジャンプし、
  そこから再生を始める
```





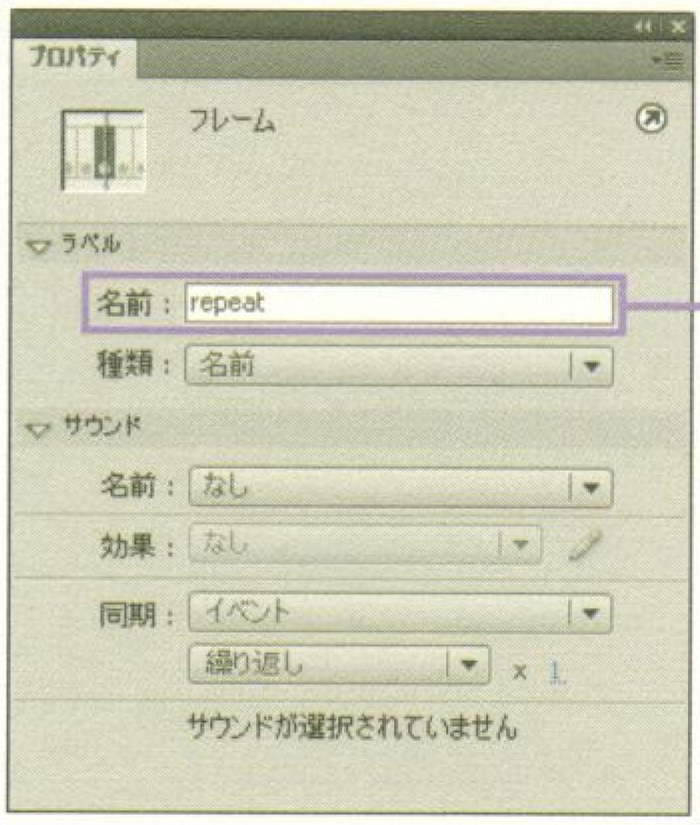
40フレーム目まで再生されると、20フレーム目にジャンプし、そこから再生が始まる

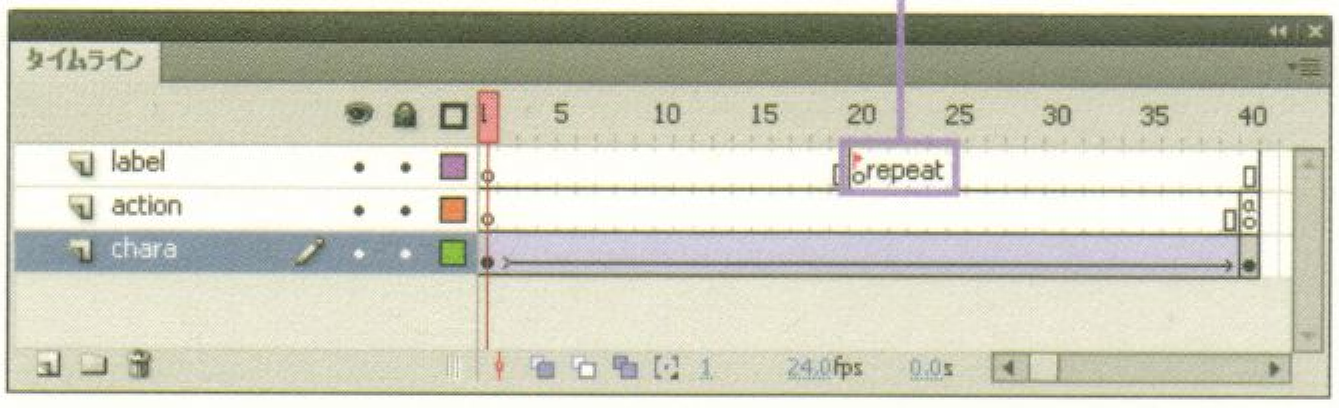
▶▶ gotoAndPlay( ) —— (2)

gotoAndPlay( )には、前ページで説明したようにジャンプ先のフレームをフレーム数で指定する以外にも、フレームのラベル名で指定する方法があります。

先程のサンプルにレイヤーを追加し、20フレーム目に空白のキーフレームを作成し、[プロパティ]パネルの「ラベル」の「名前」に任意のラベル名を入れます。

すると、タイムライン上に、「名前」で設定されたラベルが表示されます。



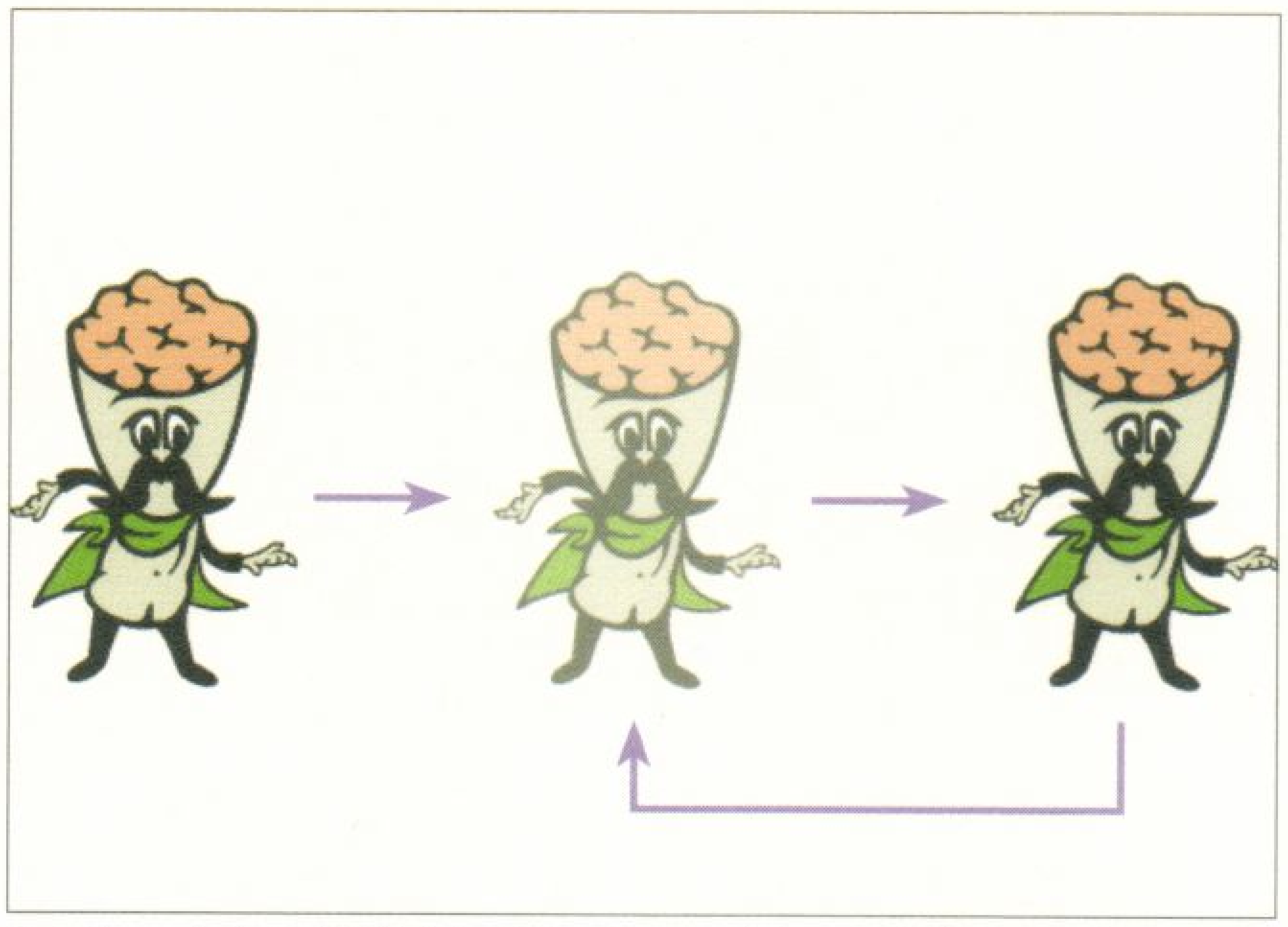


40フレーム目のスクリプトを下記のように書き換えます。

ここでムービープレビューすると、フレーム数を指定したのと同様の動作でアニメーションが再生されます。

```
gotoAndPlay("repeat");
```

⬆ ( )内に指定したラベルがついたフレームにジャンプし、そこから再生を始める



40フレーム目まで再生されると、「repeat」という名前のついたフレーム(20フレーム目)にジャンプし、そこから再生が始まる

▶▶ gotoAndStop( )

gotoAndPlay( )同様、タイムラインの制御でよく使われるのが、gotoAndStop( )です。

これは( )内で指定したフレームにジャンプし、そこで再生ヘッドを停止します。

フレームの指定は、gotoAndPlay( )同様、フレーム数でもフレームのラベル名でも指定できます。上で作ったサンプルで、gotoAndPlayをgotoAndStopに書き換え、動作を確認してみてください。

LECTURE 4-2

プログラミング［基本］

MovieClipを制御する

MovieClipを使う最大の利点は、ActionScriptで制御ができることです。ActionScriptで制御することにより、インタラクティブなコンテンツを作ることができます。

MovieClipが使われているサンプル

■ PRACTICE 3：
マウスの位置に円を表示する……… P208へ

合わせて読むと便利な項目

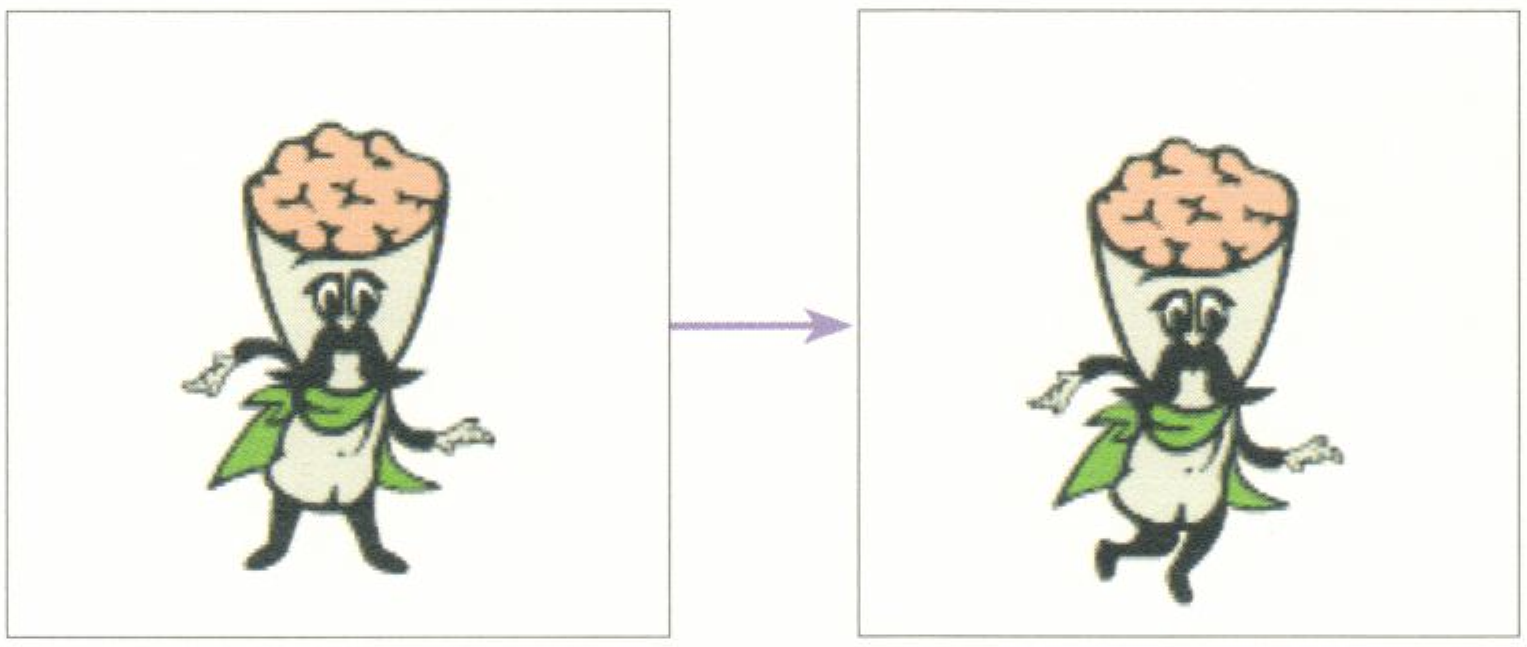
■ シンボル …………………………… P034へ
■ タイムラインを制御する…………… P080へ

ムービークリップの中のタイムラインを制御する

「1-5 シンボル」で触れたように、ムービークリップもタイムラインを持っています。ですから、タイムラインを制御（再生：playや停止：stop）するのと同じように、ムービークリップも制御できるわけです。

画面上にムービークリップが配置されています。そのムービークリップにインスタンス名「chara」と指定します。
ここでは、gotoAndPlay（フレームを指定して再生）を利用します。

```
//MovieClip(chara)の中のタイムラインを再生する(歩くアニメーション)
chara.gotoAndPlay("walk");
```



その他詳しくは、「4-1 タイムラインを制御する」を見てください。

play ( )	再生
stop( )	停止
gotoAndStop( )	フレームを指定して停止

MovieClipのプロパティ

ムービークリップをActionScriptで制御するには、プロパティを知らなくてはなりません。プロパティとは、簡単にいうと、ムービークリップのできることを指し、これを知れば、自由に制御できるようになります。

▶▶ ムービークリップの位置を変更する

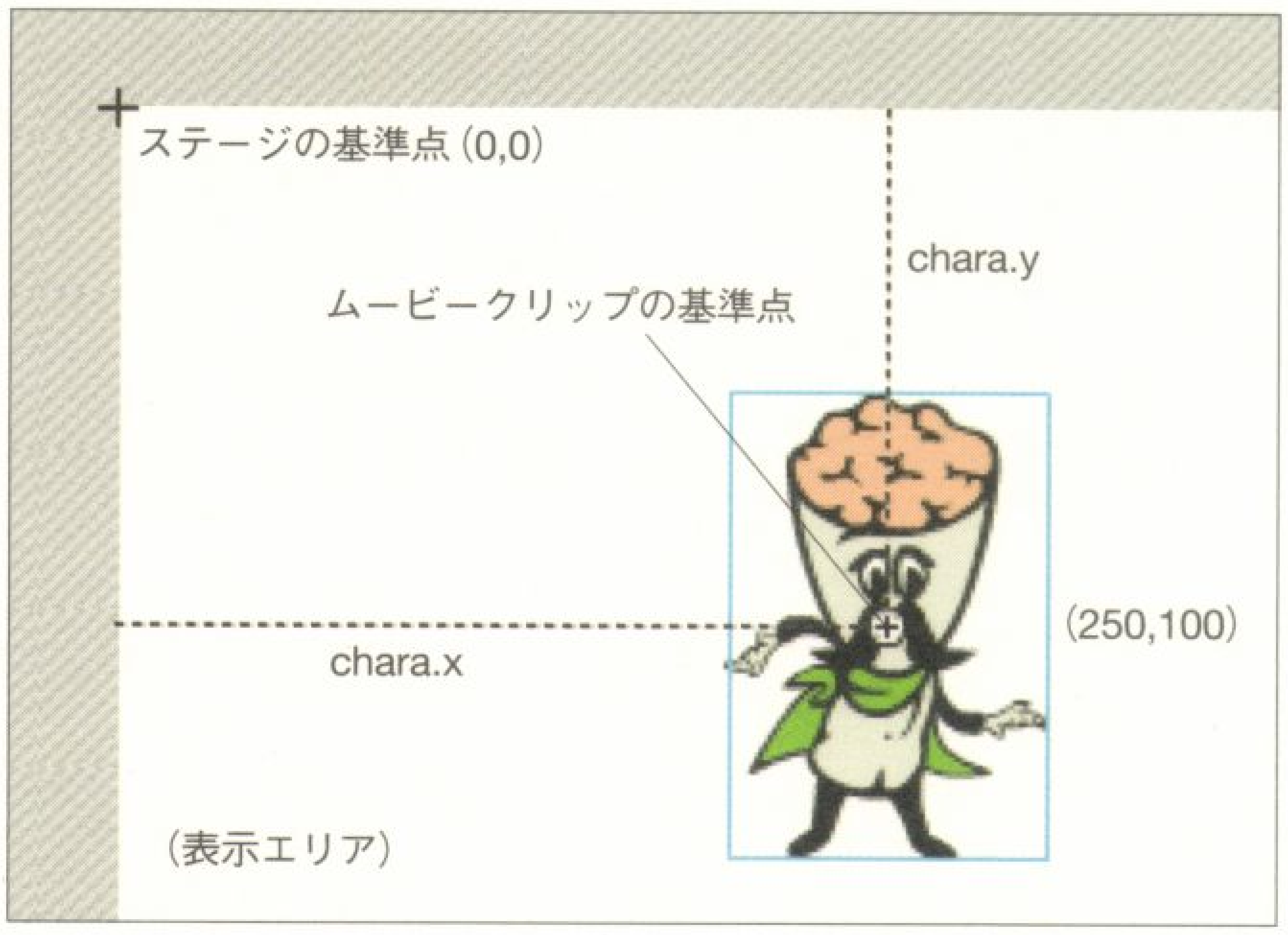
基準点を理解する

ムービークリップを制御する際には基準点がとても重要になります。
基準点とは、右図のようにムービークリップを選択した際に、表示される「+」マークの位置のことです。

位置を変更する

ムービークリップはこの点を基準として、動きます。この点を数値で指定することで、ムービークリップが変化するわけです。
位置を指定する場合には、X軸とY軸の2つを個別に指定しなくてはなりません。

x	X軸の位置を数値（基準点からのピクセル）で指定/取得
y	Y軸の位置を数値（基準点からのピクセル）で指定/取得



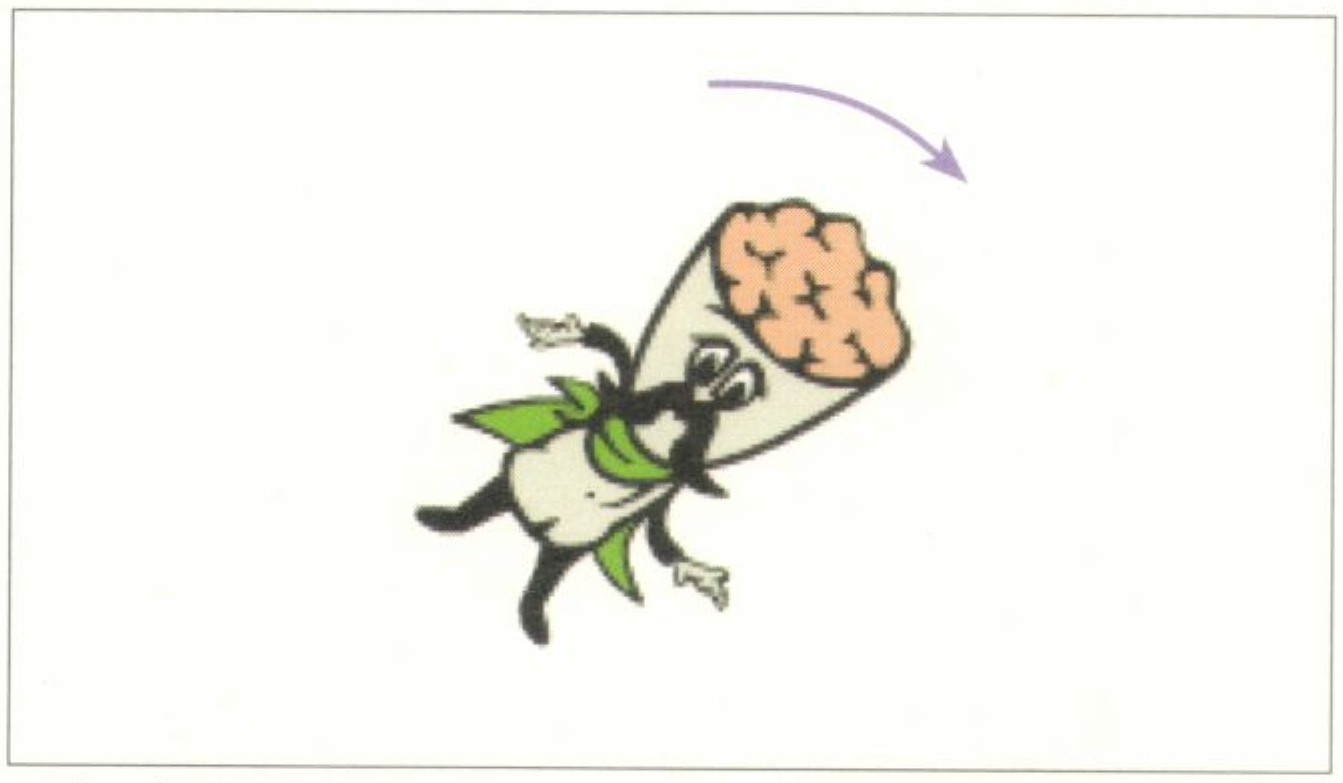

```
//MovieClip(chara)のX位置を250にする
chara.x = 250;
//MovieClip(chara)のY位置を100にする
chara.y = 100;
```

▶▶ ムービークリップを回転させる

回転も、位置の変更同様に、基準点を中心に回転します。
回転角度は数値（0〜360）で指定、数値が0以上（プラス）であれば右回転、逆に0以下（マイナス）であれば左回転になります。

rotation	角度を数値（整数）で指定/取得

```
//MovieClip(chara)を45度に回転する
chara.rotation = 45;
```



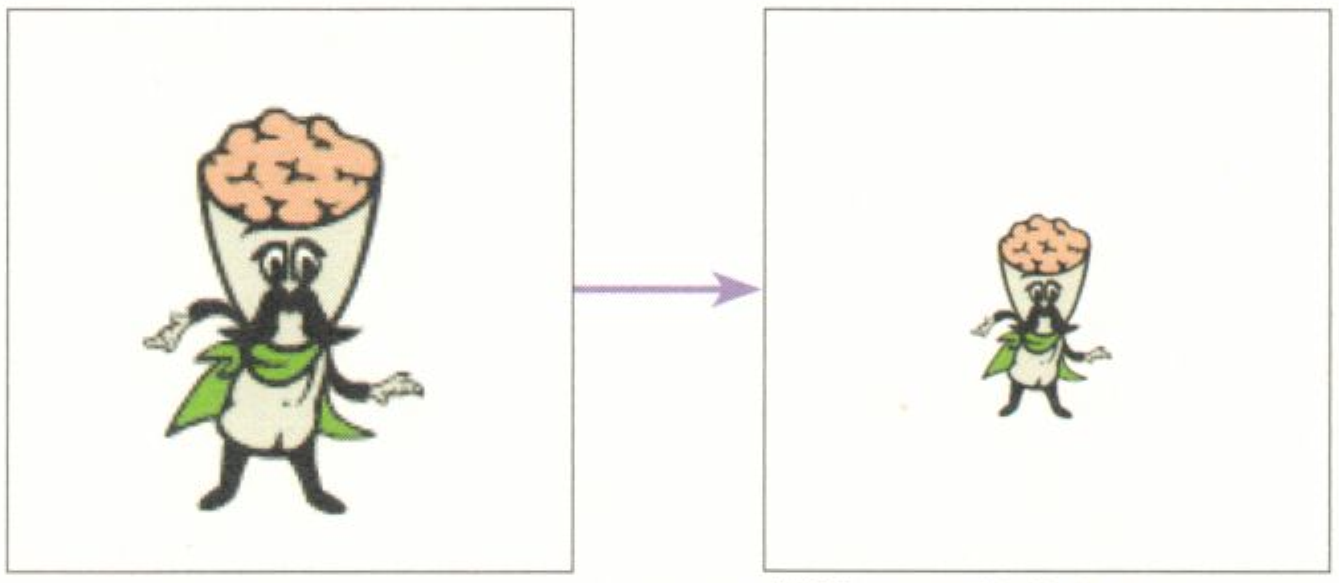
45度、右に回転する

▶▶ ムービークリップを拡大/縮小する

拡大・縮小は100%を1として、パーセンテージで指定します。等倍（デフォルト値：100%）＝1、2倍＝2、10%＝0.1となります。
また、拡大・縮小も位置同様に、横と縦を別々に指定します。

scaleX	横幅を数値（ピクセル）で指定/取得
scaleY	縦幅を数値（ピクセル）で指定/取得

```
//MovieClip(chara)の大きさ(横幅)を50%にする
chara.scaleX = 0.5;
//MovieClip(chara)の大きさ(縦幅)を50%にする
chara.scaleY = 0.5;
```



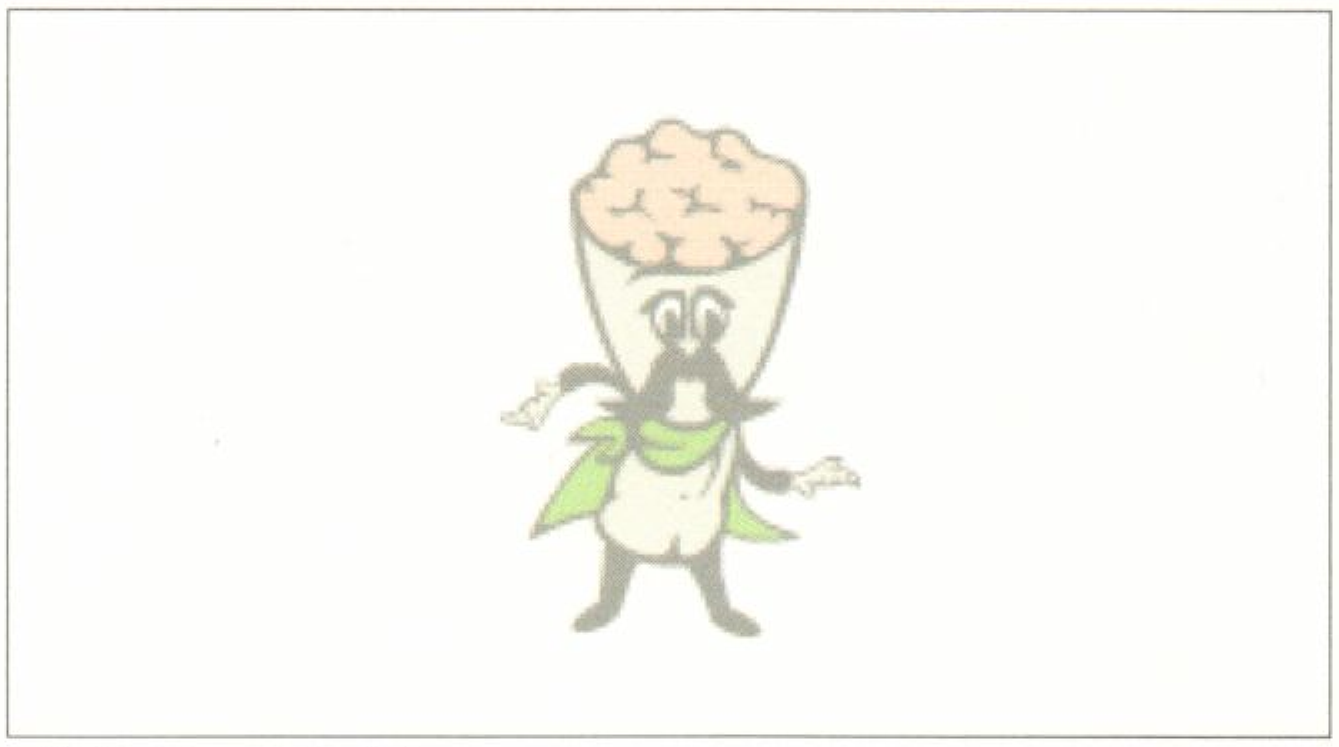
縦横50%に縮小する

▶▶ ムービークリップの不透明度を変更する

不透明度も拡大・縮小と同様で、不透明度100%を1としてパーセンテージで指定します。不透明（デフォルト値：100%）＝1、透明＝0となります。

alpha	不透明度をパーセンテージ（0〜1.0）で指定/取得

```
//MovieClip(chara)の不透明度を40%にする
chara.alpha = 0.4;
```



不透明度を40%にする

▶▶ その他のMovieClipのプロパティ一覧

currentFrame	現在のフレーム番号を取得
height	横幅を数値(ピクセル)で指定/取得
mouseX	ムービークリップ内の基準点からのマウスのX軸を数値で取得
mouseY	ムービークリップ内の基準点からのマウスのY軸を数値で取得
name	インステンス名を指定/取得
parent	1つ上の階層へアクセス
root	メインタイムラインへアクセス
totalFrames	ムービークリップの最大フレーム数を取得
visible	表示/非表示をブール値(true/false)で指定/取得
width	横幅を数値(ピクセル)で指定/取得

LECTURE 4-3

プログラミング［基本］

function

プログラムを効率的に進めるため、function（関数）を利用します。

functionが使われているサンプル

■ PRACTICE 3：
functionを利用する ……………… P205へ

合わせて読むと便利な項目

■ trace ……………………………… P071へ
■ イベントリスナ…………………… P090へ
■ 変数 ………………………………… P094へ
■ 演算子 ……………………………… P101へ
■ ENTER_FRAME …………………… P121へ

functionの考え方

functionとは何か？を簡単に「使えないバイト君とコンビニの店長」という例を出して説明します。

場所は、とあるコンビニです。店長一人だったので、新しくバイトを雇うことにしました。しかし、このバイト君はまったく気が利きません。例えば、「掃除をお願い！」と言うだけでは一人で動けません。毎回「床をモップで掃除」「窓を拭いて」「入り口をほうきで掃き」「棚を整理」など事細かに説明をしなくては動くことができません。このままでは、店長は毎回彼に掃除をさせるためにすべてを説明し続けなければなりません。あまりにも大変なので、店長はメモを作りバイト君に持たせました。メモには掃除の内容が事細かに書いてあります。そして、そのメモには「掃除内容」と書かれています。こうして店長は「掃除をお願い！」と言うだけでバイト君に事細かな掃除をお願いすることに成功しました。



毎回、事細かに説明が必要



メモ（指示書）を使う

さて、ここのどこが「function」かというと、メモに当たります。バイト君はPC（今回の場合Flash）そして、あなたは店長です。Flashは指示されたときにのみ指示の内容を実行します。つまり、何度も同じ処理をさせたい場合はメモを書く前の店長のように毎回も同じ命令を指示しなくてはなりません。こうしたプログラムのやり方では、書き忘れや、書き間違いといったミスが増えてしまいます。さらに、指示を追加・変更する際には、今まで書いたすべての命令を修正・変更しなくてはなりません。その点メモを作っておけばそのメモ1つ修正すれば良いだけですので、効率的に作業を行うことができます。
これが一般的に関数（function）と呼ばれるもので、できるだけ使いまわしが利くように作るのが目的とされています。

functionの使い方

functionの作り方と実行の仕方について、解説します。

▶▶ functionの作り方（関数を定義する）

右のコードのように、functionの中の{ }（中括弧）で囲った中にプログラムを書いていきます。

```
function 関数名(引数){

    ここに処理内容を記述      ── 処理ブロック

}
```

関数の定義は以下のような要素があります。

関数名

関数名は、その関数を呼び出すための指示書の名前です。この指示書にあたる、functionは用途に合わせていくつでも作ることができます。
そこで、的確に指示ができるように、命令に対して適切な名前を付けることも大切です.

処理ブロック

処理ブロックは、細かい指示書の内容を記述します。先ほどの掃除の指示をActionScriptに倣って書いてみましょう。

例

```
function 掃除をお願い(){
    床をモップで掃除;
    窓を拭いて;
    入り口をほうきで掃き;
    棚を整理;
}
```

▶▶ functionの実行の仕方

作ったfunctionを実行するには「function名( );」と書きます。

例

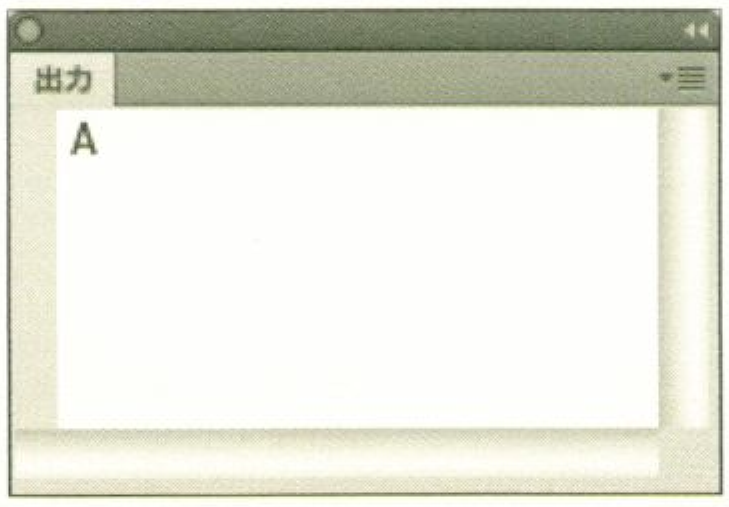
```
掃除をお願い();
```

使用例

```
//文字列「A」を出力するfunction
function traceA():void{
    trace("A");
}

//実行する
traceA();
```

出力結果



引数

functionにはさらに、「引数」と呼ばれる便利な機能があります。

▶▶ 引数とは

引数とは、functionを使い回すために、利用するfunctionの中のみで利用される変数のことです。こうすることによって汎用的に使える関数が作れます。引数は、関数へ情報を与えたい場合にのみ使用します。

```
function 関数名(引数){

    ここに処理内容を記述      ── 処理ブロック

}
```

例えば、「棚の整理」ですが、コンビニにはいろいろな商品の棚があります。雑誌の棚や飲料水の棚などをその都度指定することができれば、この引数を利用することで、「掃除と一緒に雑誌の棚も片付けといて！」など汎用的な指示を簡単に出すこともできます。場所の指示の仕方は右の例のようになります。このように片付けてほしい場所を指定したい時などに有効です。

例

```
function 掃除をお願い(場所){
    床をモップで掃除;
    窓を拭いて;
    入り口をほうきで掃き;
    [場所]棚を整理;
}
```

括弧の始まりと終わりが分かりやすいように、処理ブロック中のコードの先頭には「タブ」を入れて表記すると良いでしょう。

```
function 関数名(引数){
(タブ)ここに処理内容を記述    ── 処理ブロック
}
```

▶▶ 引数を使った際の実行方法

引数により、「場所」という変数へ「本棚」が渡り、結果、掃除と同時に[雑誌]棚の整理という指示を出すことができます。

例

```
掃除をお願い(雑誌);
```

使用例

```
//好きな文字列を出力するファンクション
function traceAnyText(str:String):void{
    trace(str);
}

//実行する
traceAnyText("A");
```

出力結果



引数は、複数設定することも可能です。
こうして、複数の引数を指定する際は、「,(カンマ)」区切りで指定します。これにより、より柔軟な対応のできる指示書を作ることができます。

```
function ファンクション名(引数1, 引数2){
    処理内容
    [引数1]を使った処理内容
    [引数2]を使った処理内容
}
```

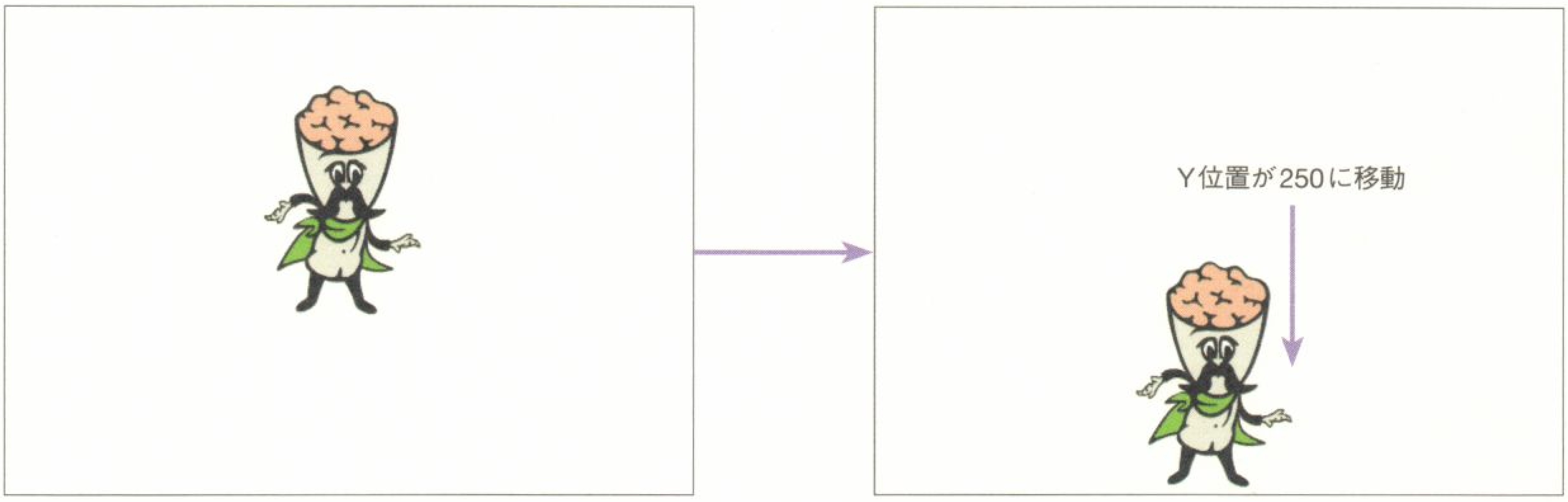
▶▶ サンプル：ムービークリップと合わせて使ってみよう

Y位置を移動する

ステージ上に配置したムービクリップ(インスタンス名：chara)のY位置を「250px」に移動するファンクションを作ってみましょう。

```
//ムービークリップ(chara)のY位置を250にする
function setPos():void{
    chara.y = 250;
}

//実行する
setPos();
```



引数を使ってY位置の値を指定する

また、Y位置の値をファンクション実行時に自由に設定できるように、引数を使って書き換えてみます。
こうしたフレキシブルなファンクションを作ると、似通ったプログラムを何度も作らずに済みます。

```
//ムービークリップ(chara)のY位置を自由に設定できる
function setPos(num:int):void{
    chara.y = num;
}

//Y位置250に設定して実行する
setPos(250);
```

LECTURE 4-4

プログラミング［基本］

イベントリスナ

「3-6 イベントの概念」では、「ボタンをクリックしたら関数を実行する」という例を使ってイベントの考え方を説明しました。ここでは、実際にサンプルを作りながらActionScriptを使ったイベント処理について学んでいきます。

イベントリスナが使われているサンプル

- PRACTICE 2：ボタンの動きをスクリプトで制御する …………………………… P196へ
- PRACTICE 5：サムネイルに画像を読み込む ……… P226へ
- PRACTICE 8：ウェブカメラの起動と制御 ………… P274へ

合わせて読むと便利な項目

- ActionScriptとは…………………………… P064へ
- イベントとは ……………………………… P078へ

イベントリスナを使ったイベント処理

イベントリスナを使ってイベント処理を行うには、addEventListener( )メソッドを使います。

▶▶ ボタンの準備

「3-6 イベントの概念」の最後では、下の図を使ってイベント処理に必要な要素をまとめました。

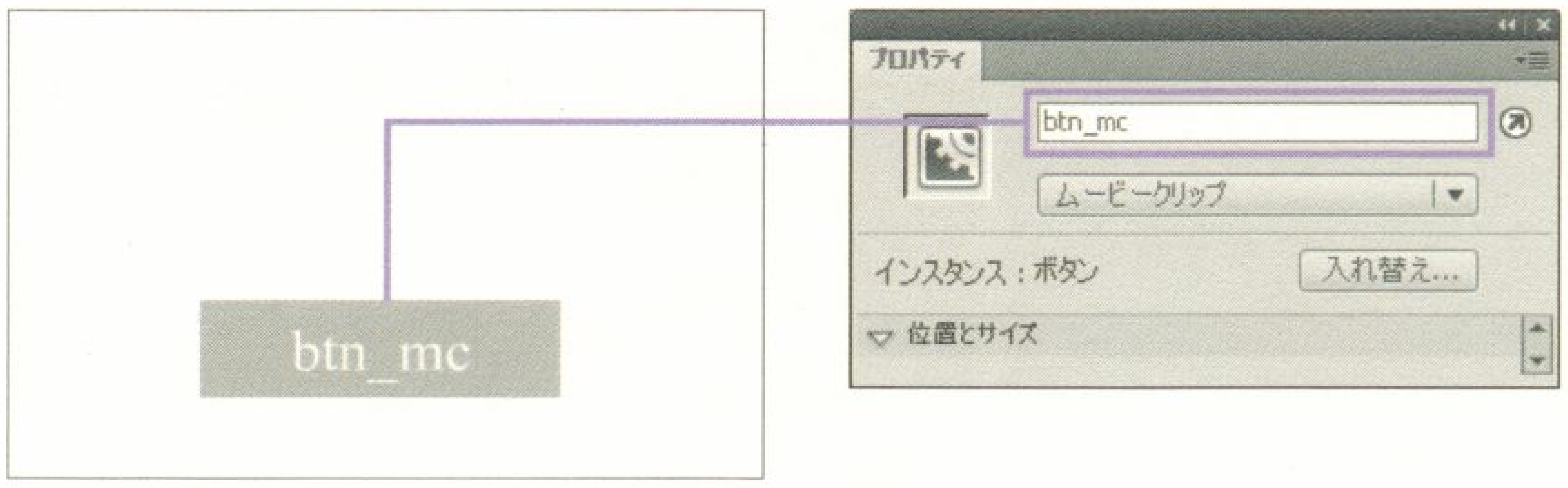
イベント処理に必要な要素

- ・イベントのターゲットインスタンス（誰が）
- ・イベントの種類（○○したら）
- ・リスナー関数（××する）

「ボタンをクリックしたら、関数eventTest( )を実行する」

ターゲット　イベントの種類　リスナー関数

これを実際にFlash上で動作させてみましょう。
まずは前準備としてFlash上でボタンを作成しておきます。ステージ上に矩形ツールを使って長方形を書き、その上で右クリックで［シンボルに変換→ムービークリップ］を選択し、インスタンス名を「btn_mc」としておきます。



▶▶ addEventListener( )メソッドでボタンを動作するようにする

まずはボタンをクリックされたら実行されるリスナー関数を作ります。

```
//リスナー関数を作る
function eventTest(event){
   trace("ボタンがクリックされました。");
}
```

次に、イベントリスナを使って、ボタン(ターゲットとなるインスタンス)と関数を結び付けます。

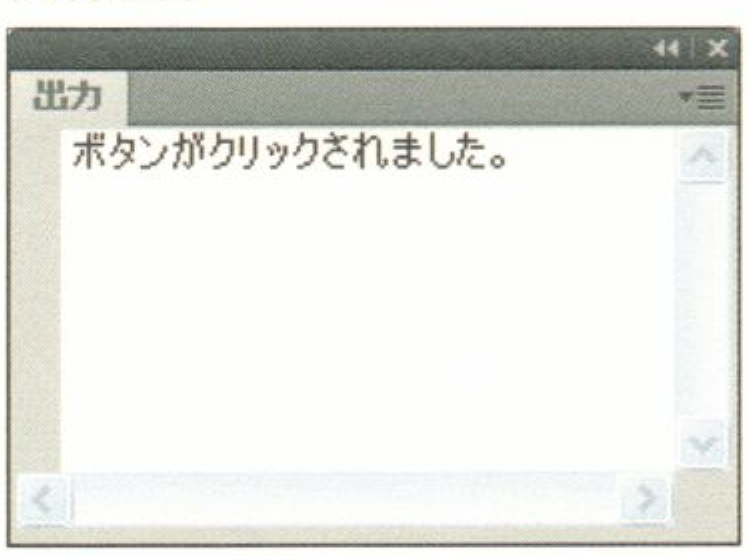
```
//イベントリスナでイベントと関数を結び付ける
btn_mc.addEventListener(MouseEvent.CLICK, eventTest);
```

使い方

ターゲットのインスタンス名.addEventListener(イベントの種類, リスナー関数);

この状態でプレビューしてみましょう(メニューバーの［制御→ムービープレビュー］)。
ボタンをクリックすると、［出力］パネルに下のように表示されるはずです。クリックするたび関数が呼ばれるので、ボタンを連打すると、その回数分だけ［出力］パネルに文字が表示されます。

実行結果



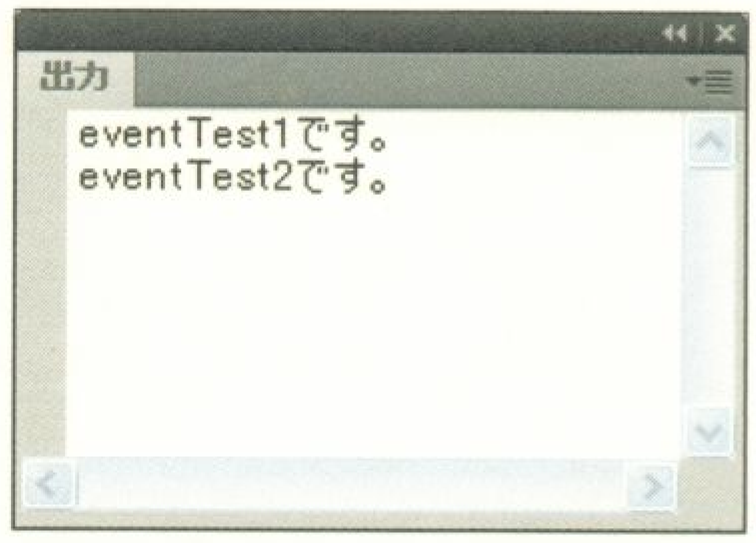
▶▶ 複数の関数を実行させる

イベントリスナを複数置くことで、1つのイベントに複数の関数を結び付けることもできます。

```
//リスナー関数を作る
function eventTest1(event){
  trace("eventTest1です。");
}
function eventTest2(event){
  trace("eventTest2です。");
}

//イベントリスナでイベントと関数を結び付ける
btn_mc.addEventListener(MouseEvent.CLICK, eventTest1);
btn_mc.addEventListener(MouseEvent.CLICK, eventTest2);
```

実行結果



▶▶ イベントリスナを削除する

一度結び付けたイベントと関数を解除するには、removeEventListener( )メソッドを使って、イベントリスナを削除することができます。

```
//イベントリスナを削除する
btn_mc.removeEventListener(MouseEvent.CLICK, eventTest1);
```

使い方

ターゲットのインスタンス名.removeEventListener(イベントの種類, リスナー関数);

▶▶ イベントの種類

前記のサンプルでは「クリック」がイベントでしたが、イベントの種類は他にも「マウスカーソルが動く」「キーボードでキーが押される」「ファイル読み込みが完了する」「再生ヘッドがフレームに来る」など、色々なものをイベントとして使うことができます。以下にイベントの例を挙げておきます。
ちなみに、イベントの種類はActionScriptで決められており、イベント定数(定数は変更できない値を表す)と呼ばれます。

イベントの例

イベント	イベントが発生するタイミング
MouseEvent.MOUSE_DOWN	マウスのボタンが押されたら
MouseEvent.MOUSE_UP	マウスのボタンが離されたら
MouseEvent.CLICK	マウスのボタンがクリック(押して離す)されたら
MouseEvent.DOUBLE_CLICK	マウスのボタンがダブルクリック(押して離すを2回)されたら
MouseEvent.MOUSE_MOVE	マウスカーソルが動いたら
KeyboardEvent.KEY_DOWN	キーボード上で何かのキーが押されたら
Event.ENTER_FRAME	再生ヘッドがフレームに来たら
Event.COMPLETE	外部ファイルの読み込みが完了したら

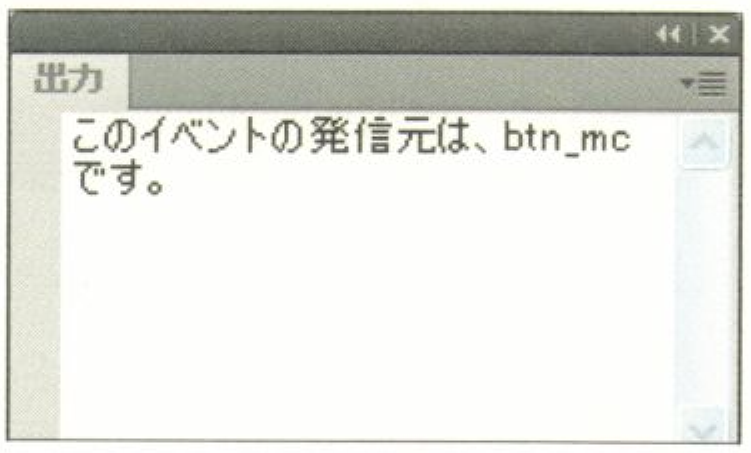
▶▶ イベント発信元を知るには

リスナー関数の中でイベント発信元のオブジェクトを指すには、Eventオブジェクトのtargetというプロパティを使います。

```
//リスナー関数を作る
function eventTest(event){
   trace("このイベントの発信元は、"+event.target.name+"です。");
}
⬆ event.target.nameでイベントの発信元の名前を取得できる

//イベントリスナでボタンを関数と結び付ける
btn_mc.addEventListener(MouseEvent.CLICK, eventTest);
```

実行結果



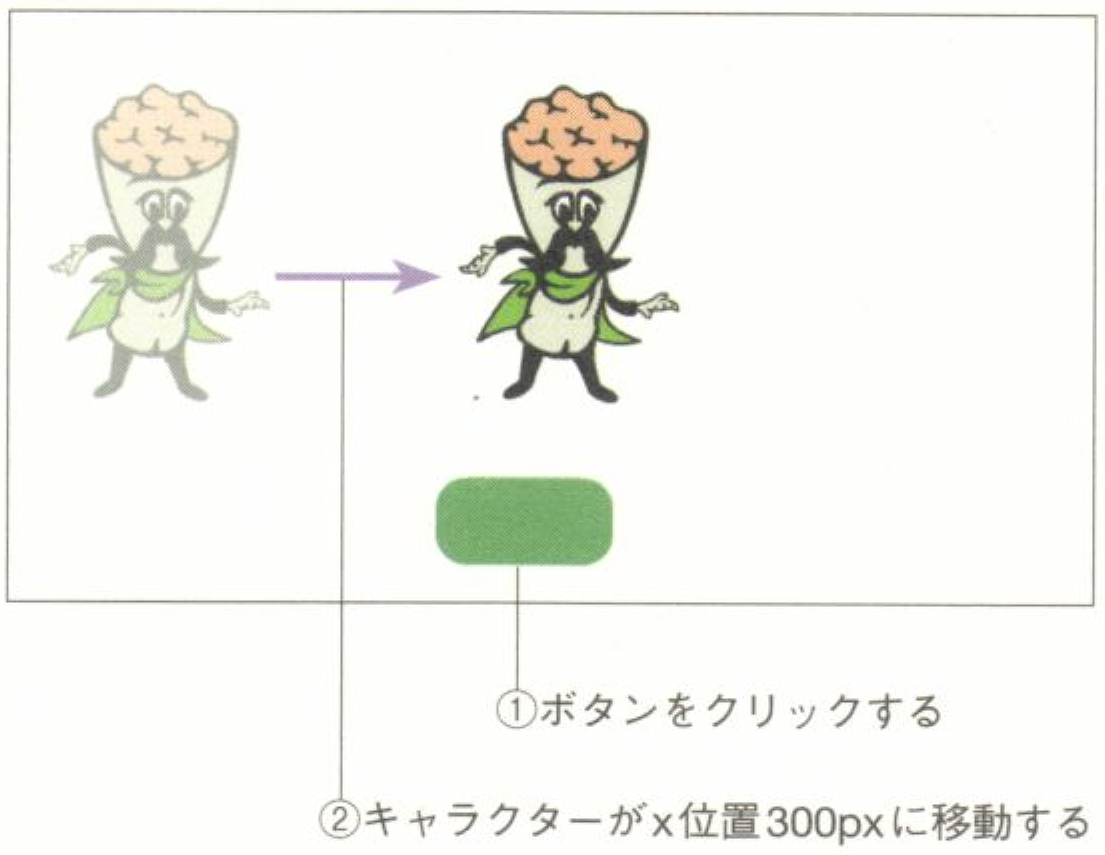
▶▶ サンプル：ボタンを押したらキャラクターの位置を動かす

ボタンとイベントリスナを使って、「ボタンをクリックしたらキャラクターの位置を動かす」という簡単なサンプルを作ってみましょう。メインタイムラインの1フレーム目に「action」、「chara」、「button」という3つのレイヤーを作り、「chara」レイヤーにキャラクターを配置し、インスタンス名を「chara」とします。同様に「button」レイヤーにボタンを配置し、インスタンス名を「btn_mc」としておきます。
「action」レイヤーに下記のスクリプトを記述します。
実行して、ボタンのクリックでキャラクターが300ピクセルのx位置に移動すれば成功です。

```
//リスナー関数を作る
function moveMisoda(event){
   chara.x = 300;
}

//ボタンを関数と結び付ける
btn_mc.addEventListener(MouseEvent.CLICK, moveMisoda);
```

実行結果





LECTURE 4-5

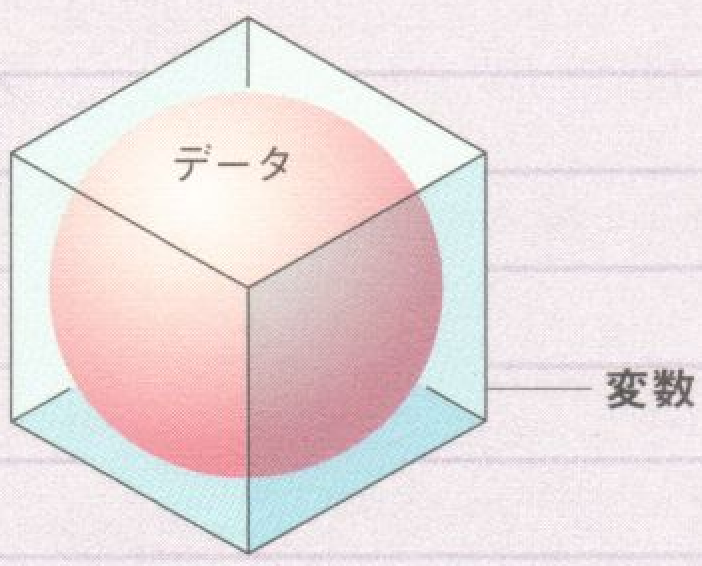
プログラミング［基本］

変数

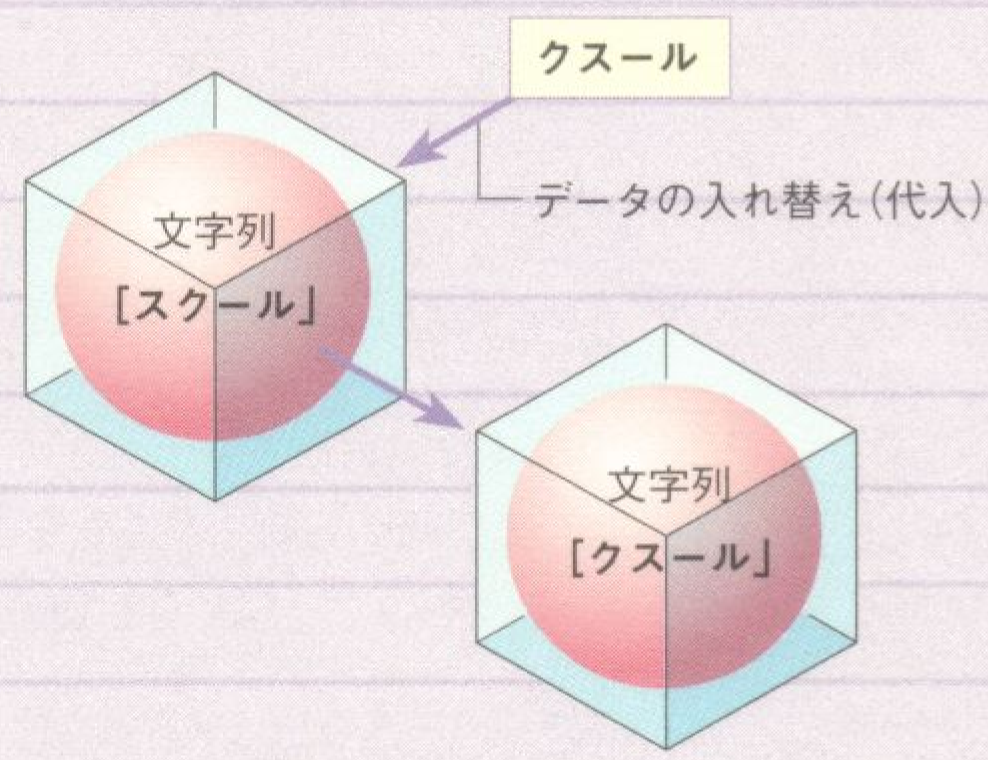
変数は、ある情報（データ）を保存、受け渡しをするためのデータ形式で、プログラミングの世界では、よく箱に例えられます。プログラミングでは、1つのステートメント（処理）が終わると、そのステートメントで使われていたデータは破棄されてしまいますが、変数はあとで他の場所でも同じデータを参照して別の処理ができるようにするための仕組みです。後ほど説明する配列もこの変数の一種と言えます。プログラミングでは必須の知識の1つです。

変数の特徴

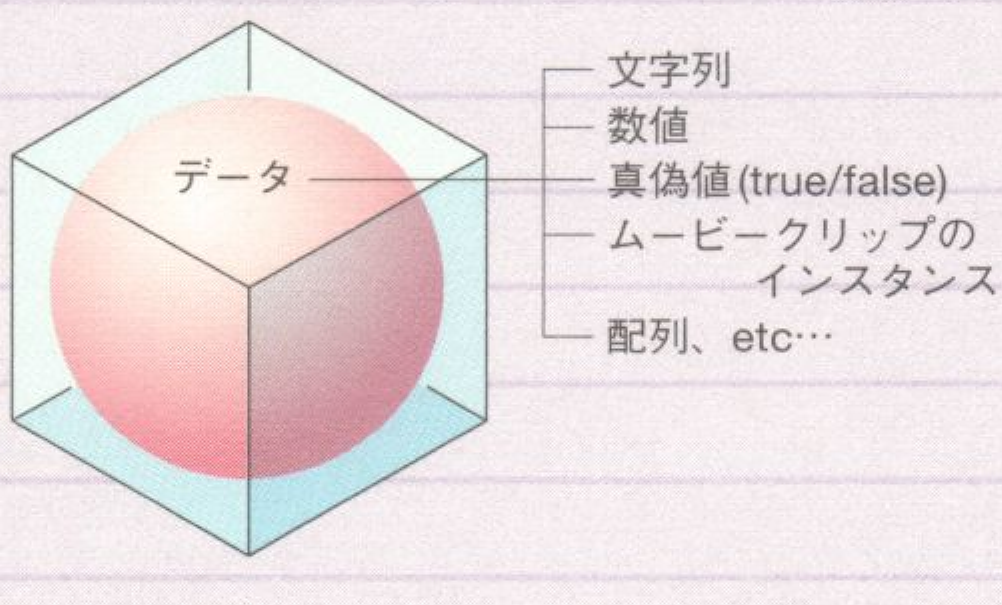
①変数はデータを記憶しておく箱



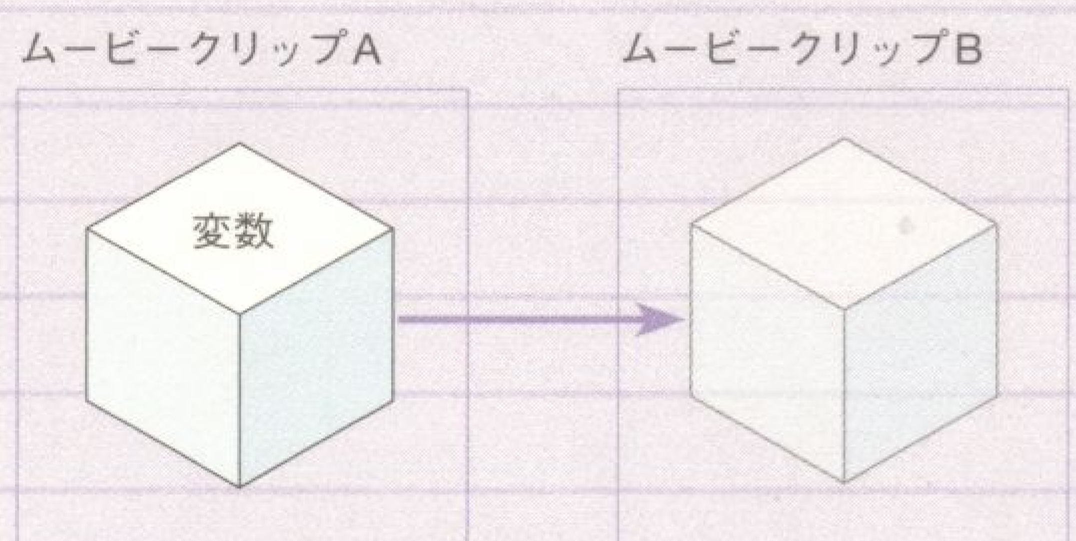
③データを入れ替えたり、データの追加 / 削除ができる



②色々なものを値として保存できる



④変数の形でデータの受け渡しができる



変数が使われているサンプル

■ PRACTICE 2：メインイメージの動きをスクリプトで制御する …………… P197へ
■ PRACTICE 5：画面に要素を配置する …………… P223へ
■ PRACTICE 6：画面に要素を配置する …………… P233へ

合わせて読むと便利な項目

■ 配列 ………………………………… P115へ

変数の基本的な使い方

それでは実際に変数を使ってみましょう。変数を使うには、大きく分けて、変数を作成（宣言）し、中に入れるものの形（データ型）を決め、値を入れる（代入）という3つのステップがあります。

▶▶ 変数の宣言

変数を作成することを、「変数を宣言する」と言います。変数を宣言するには、varを使います。
データ型とは変数の中に入れるもののタイプを指定するもので、変数名の後にコロンで指定します。

使い方

```
var 変数名:データ型;
```

使用例

```
var schoolName:String;
⬆ データ型がString(文字列)の変数schoolNameを宣言
```

▶▶ 数値を表示させる

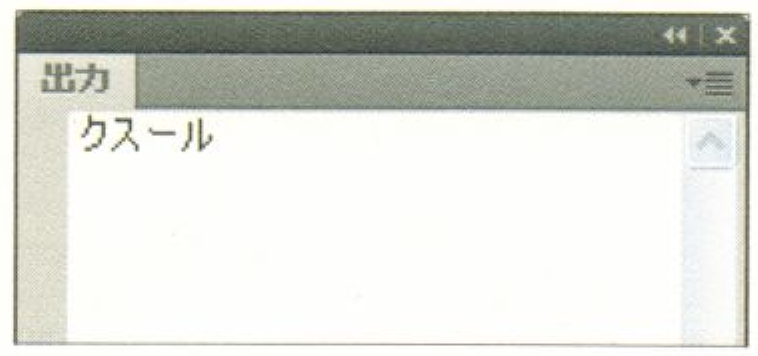
変数に値を保存させることを「代入する」と言います。変数に値を代入するには、「=」を使います。

使用例

```
var schoolName:String;

schoolName = "クスール";
⬆ 変数schoolNameに文字列「クスール」を代入
trace(schoolName); ⬅ 変数名で値を取り出すことができる
```

実行結果

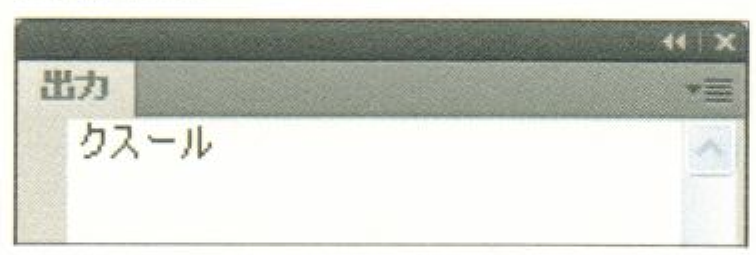


変数の宣言時に値を代入しておくこともできます。

使用例

```
var schoolName:String; = "クスール"
⬆ 変数の宣言と値の代入を同時に行う
trace(schoolName);
```

実行結果



変数の値はプログラムの途中で書き換えることができます。

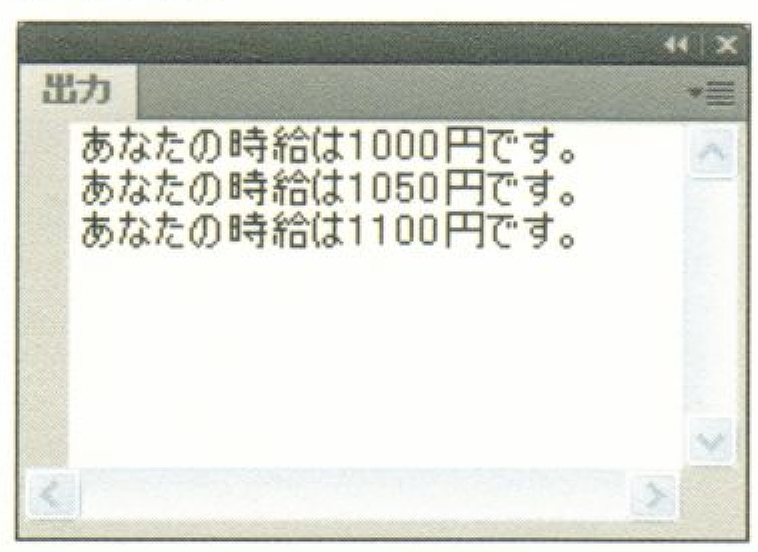
使用例

```
var jikyuu:uint = 1000;
⬆ データ型が整数(uint)の変数jikyuuを宣言し、数値1000を代入
trace("あなたの時給は" + jikyuu + "円です。");

jikyuu = 1050; ⬅ jikyuuに1050を代入
trace("あなたの時給は" + jikyuu + "円です。");

jikyuu = 1100; ⬅ jikyuuに1100を代入
trace("あなたの時給は" + jikyuu + "円です。");
```

実行結果



変数を覚えるときにつまづくのが、この「＝」を使って「代入する」という考え方です。「＝」を使っているので、a＝bが「aとbは等しい」と考えてしまいそうですが、プログラム上では「aにbを代入する」という意味になります。プログラミングでは「＝」を代入演算子と呼んでいます。

データ型と型指定

変数の宣言の際にデータ型を指定しておくことで、変数に保存できるデータの形式を決めておくことができます。ここでは、データ型の種類と使い方を見ていきます。

データ型の種類

データ型の種類には、主に右のようなものがあります。

データ型の例

データ型	初期値	値の形式
String	null	文字列
int	0	整数（-2,147,483,648～2,147,483,647（両端含む））
uint	0	正の整数（0～4,294,967,295（両端含む））
Number	NaN	整数、符号なし整数、および浮動小数点数
Boolean	false	真偽値（true,false）のみ
Array	undefined	配列
MovieClip	null	ムービークリップのインスタンス

型指定

変数の宣言の際にデータ型を指定することを、「型指定する」と言います。型指定をしておくと、プログラムの処理が若干早くなること、期待している型ではないデータが代入された際にエラーを返してくれるというメリットがあります。

使用例

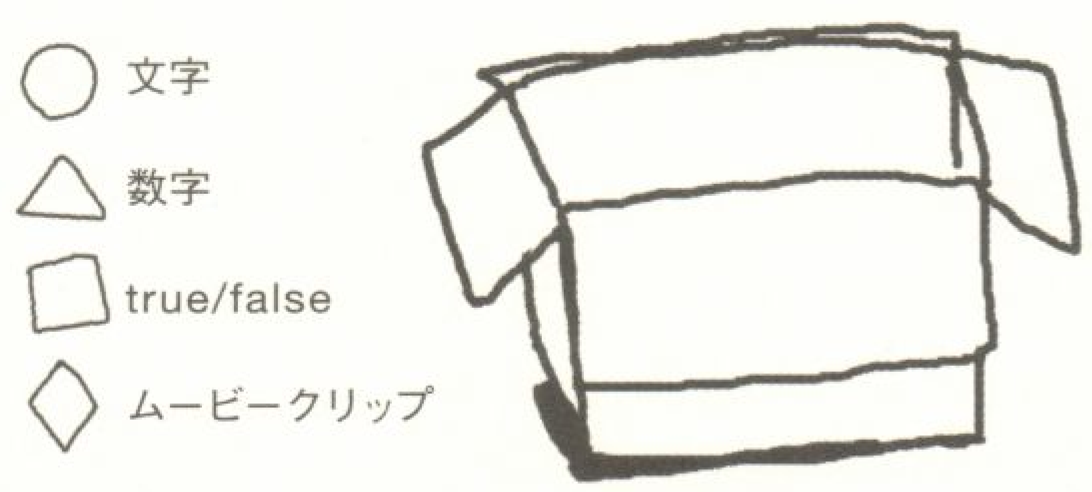
```
var schoolName:String = "クスール";
⬆ String(文字列)型の変数schoolNameを宣言し、「クスール」を代入
schoolName = 1000; ⬅ 数値を代入しようとしたため、エラーが出る
```

逆に、型指定の際にコロンの後ろに*を指定、または型指定を行わないで変数を宣言することもできます。どんな型のデータでも代入できる一方、プログラムの処理が若干遅くなります。

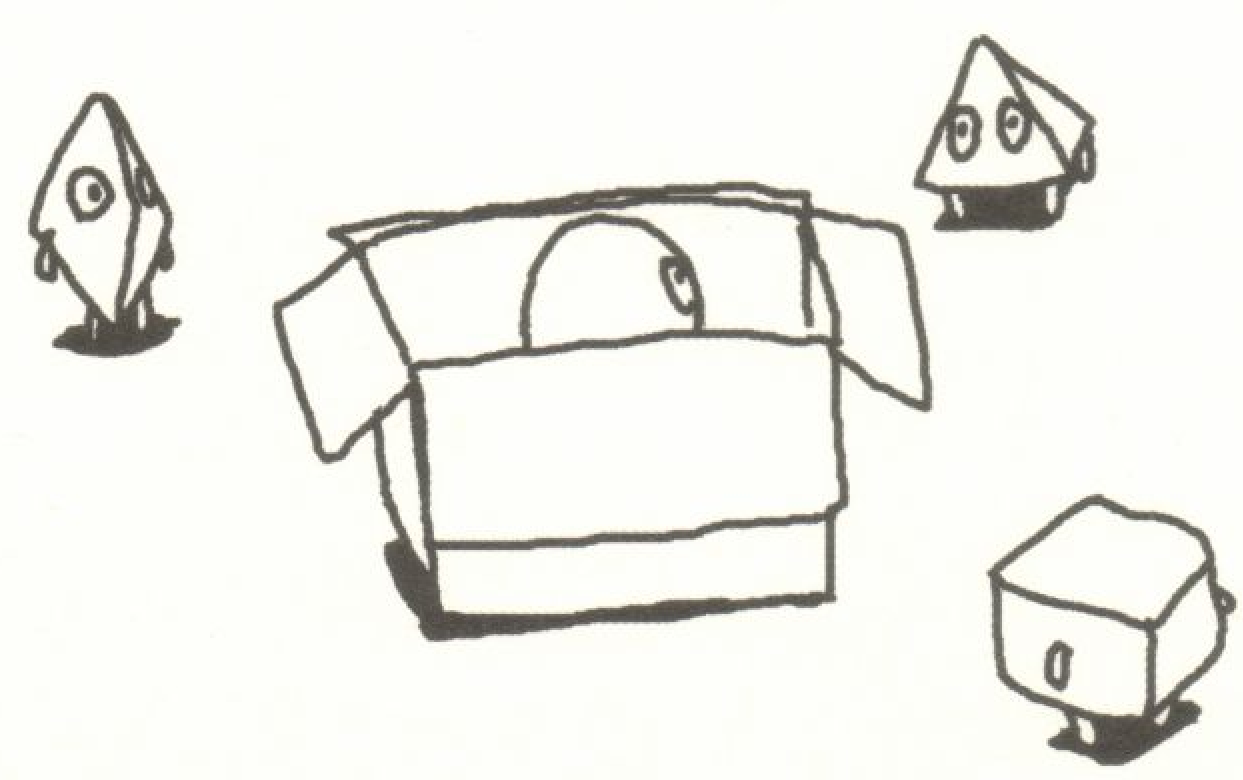
使用例

```
var schoolName; ⬅ データ型の指定しない変数schoolNameを宣言
var schoolName:*
⬆ コロンの後ろに*を付けても同じ(変数にどんなものが入るか分からない状態)
```

①変数はデータを記憶しておく箱



②色々なものを値として保存できる



データ型を変換する(キャスト)

プログラミングをしていると、数値としてとして扱っているデータを文字列に変えたり、文字列として扱っていたデータを数値に変換したい、という場面が出てきます。このようなときに使うのが型変換(キャスト)です。

▶▶ 文字列に変換する

数値を文字列に変換するには、String( )関数を使います。

使用例

```
trace(10 + 100); //数値データなので、計算結果が表示される
trace(String(10) + String(100)); //数値を文字列に変換し、連結させた
```

▶▶ 数値に変換する

文字列を数値に変換するには、右表の関数が使用できます。

文字列を数値に変換する関数

関数	変換後のデータ型	備考
int( )	int	
uint( )	uint	
Number( )	Number	
parseInt( )	Number	文字列を整数値に変換
parseFloat( )	Number	文字列を浮動小数点に変換

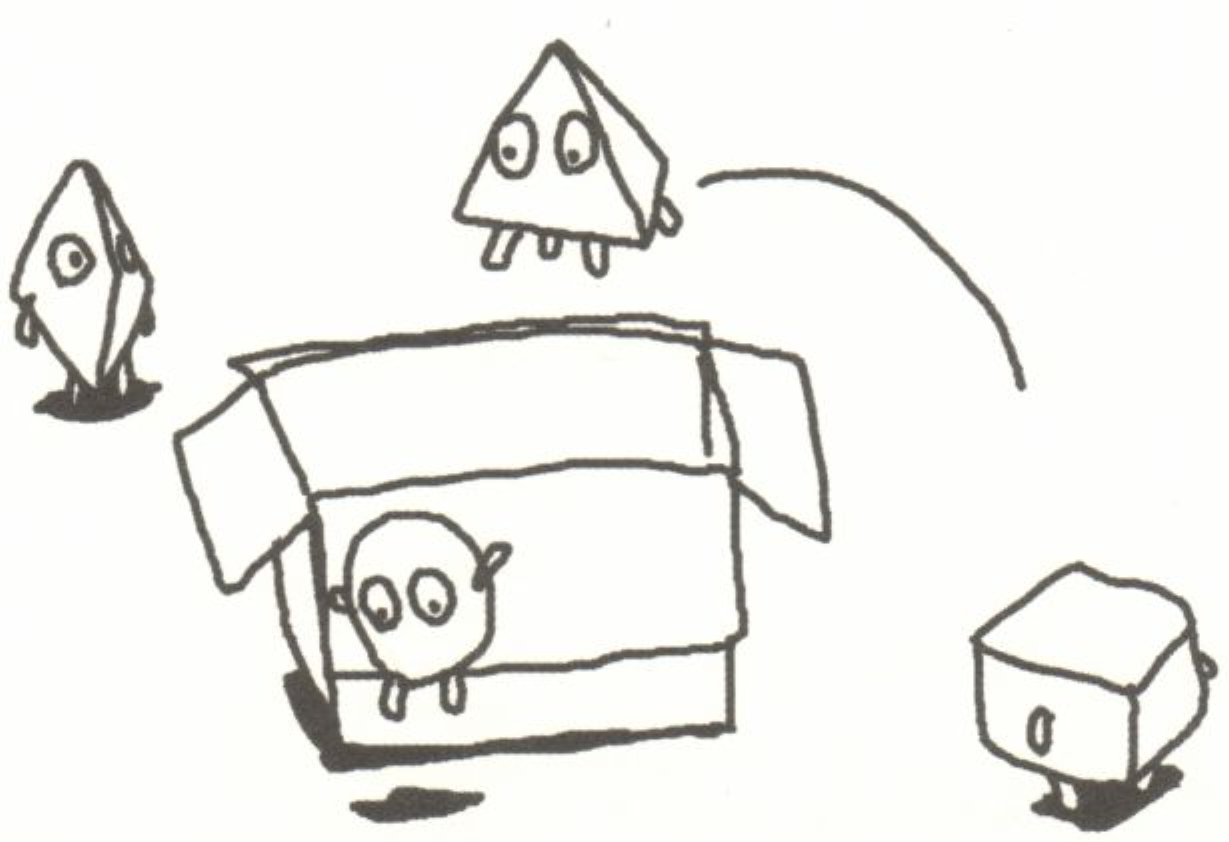
使用例

```
trace("100" - 10); ⬅ 文字列ー数値の計算式なので、エラーになる
```

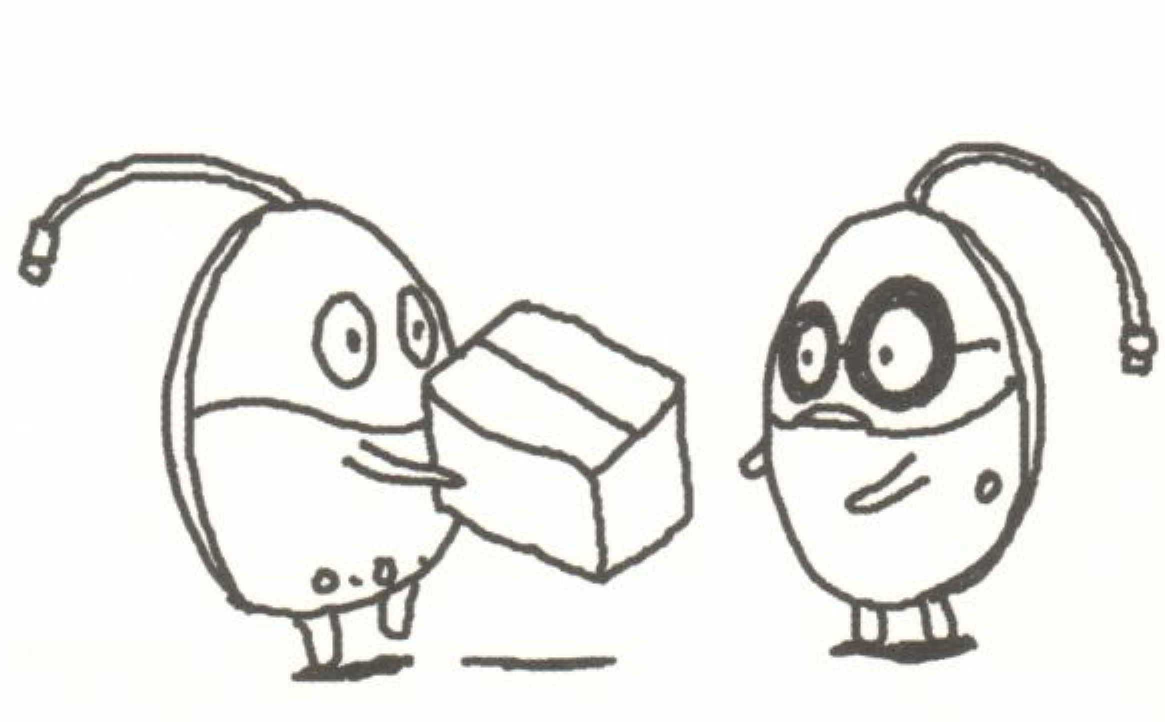
使用例

```
trace(uint("100") -10); ⬅ 文字列を数値に変換したので、計算が可能になった
```

③データを入れ替えたり、データの追加／削除ができる



④変数の形でデータの受け渡しができる



LECTURE 4-5

変数の有効範囲（スコープ）

変数はどこに書いてもどこからでもその情報を取り出せる（参照できる）わけではなく、変数を宣言した場所によってその情報を参照できる範囲、つまり有効範囲が決まります。この有効範囲のことを、スコープと言います。
変数は宣言した場所によりグローバル変数とローカル変数に分けられ、2つの変数のスコープは異なります。

▶▶ グローバル変数

コード内すべての範囲で参照できる変数をグローバル変数と言います。これは関数の外に書かれた変数で、ActionScript 3.0では、varを使って宣言する必要があります。

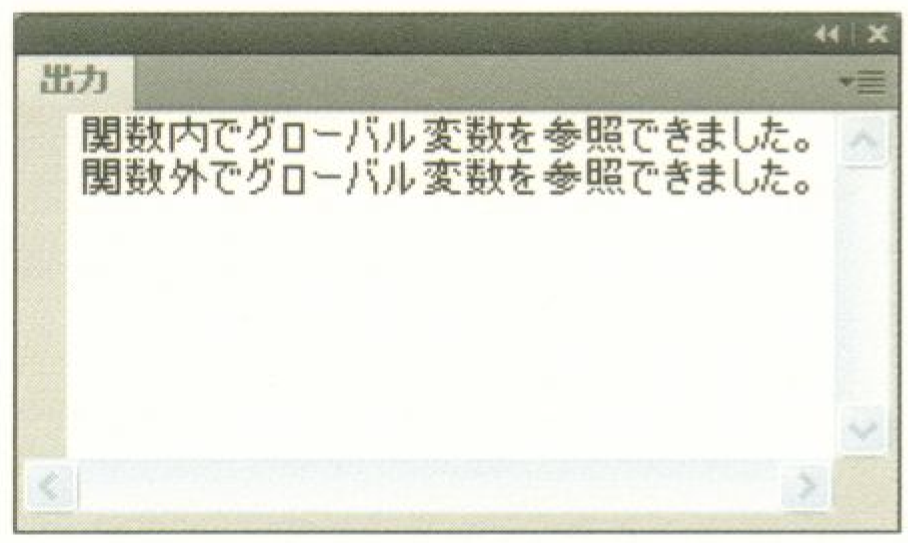
使用例

```
var globalString:String = "グローバル変数";

function checkScope(){
    trace("関数内で" + globalString + "を参照できました。");
}

checkScope();
trace("関数外で" + globalString + "を参照できました。");
```

実行結果



▶▶ ローカル変数

関数の中に書かれた変数をローカル変数と言います。これはグローバル変数と違って、スコープがその変数が書かれた関数内のみとなっています。

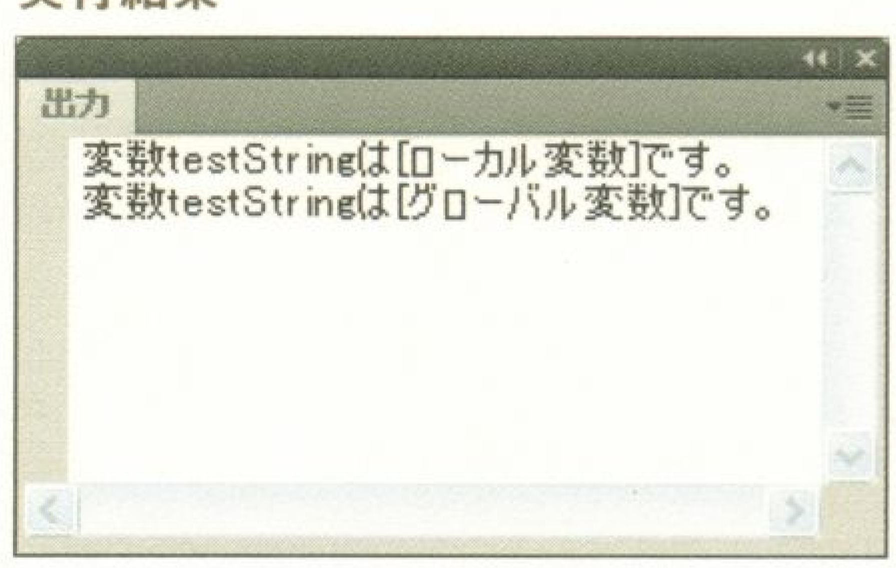
使用例

```
var testString:String = "[グローバル変数]"; ← 関数の外で変数を宣言

function checkScope(){
    var testString:String = "[ローカル変数]"; ← 関数の中で変数を代入
    trace("変数testStringは" + testString + "です。");
}

checkScope();
trace("変数testStringは" + testString + "です。");
```

実行結果



COLUMN

ローカル変数とブロックスコープ

ActionScript 3.0の変数では、C++やJavaで使えるブロックスコープが存在しません。あくまでローカル変数のスコープは関数単位なので、C++やJavaのようなブロック内だけで有効な変数を宣言することができません。
C++、Javaの経験者はここでハマってしまうことがあるので、注意が必要です。

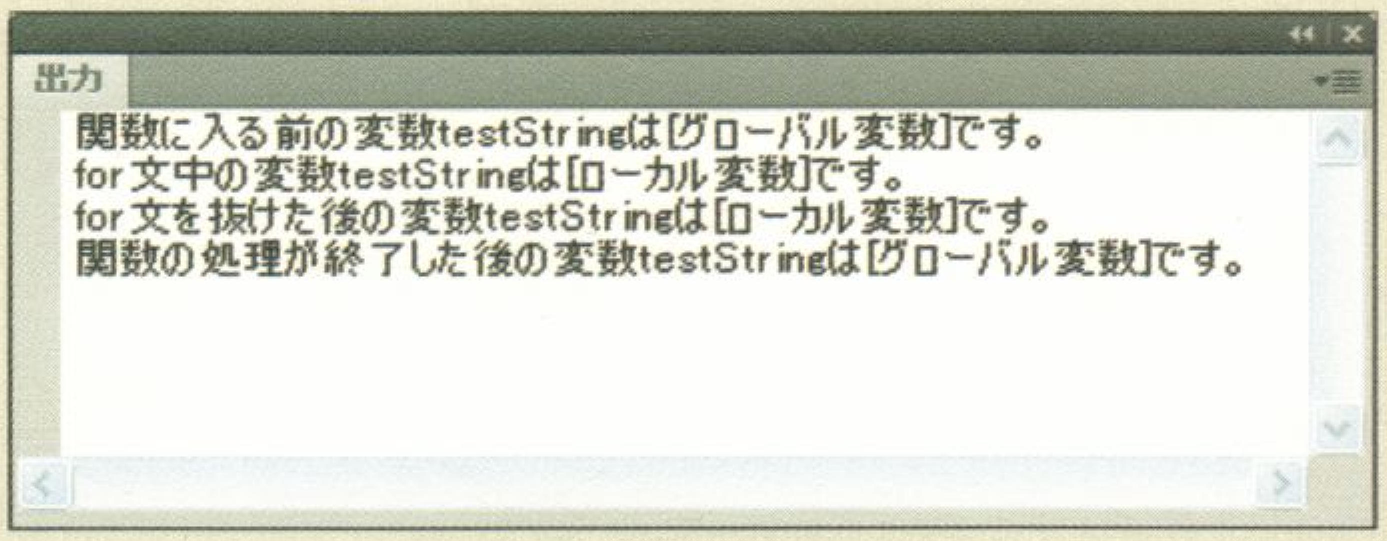
使用例

```
var testString:String = "[グローバル変数]";
trace("関数に入る前の変数testStringは" +testString+ "です。")

function checkScope(){
    for(var i = 0; i < 1; i++){
        var testString:String = "[ローカル変数]";
        trace("for文中の変数testStringは" + testString + "です。");
    }
    trace("for文を抜けた後の変数testStringは" +testString+ "です。");
}

checkScope();
trace("関数の処理が終了した後の変数testStringは"+ testString + "です。");
```

実行結果



COLUMN

文字列、数値、変数の扱い方

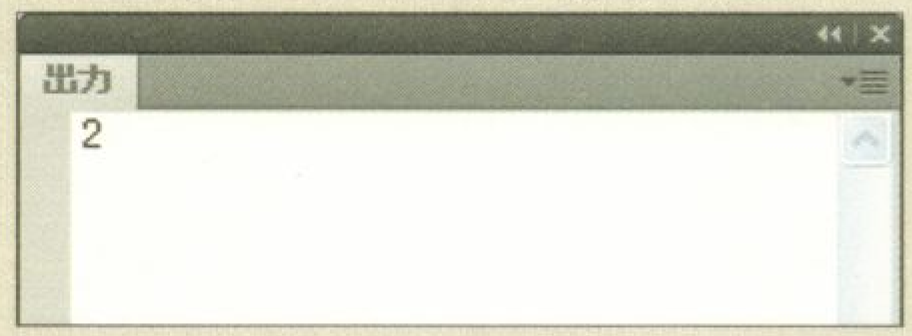
プログラミングで変数を扱う際、扱っているものが数値なのか文字列なのか、というところで混乱することがよくあります。型変換(キャスト)については本文で解説していますが、数値、文字列、変数を組み合わせたときにどうなるか、という挙動を知っておくことも重要です。
ここでは、文字列、数値、変数の組み合わせのパターンを、簡単なサンプルを使って見ていきます。
あまり詳しい解説はしないので、コードと実行結果を見ながら、どうしてこのような結果になるのか、考えてみましょう。

▶▶ 文字列と数値を足す

```
var a = 1; ⇐ 変数aは数値
var b = 1; ⇐ 変数bは数値

trace(a+b);
```

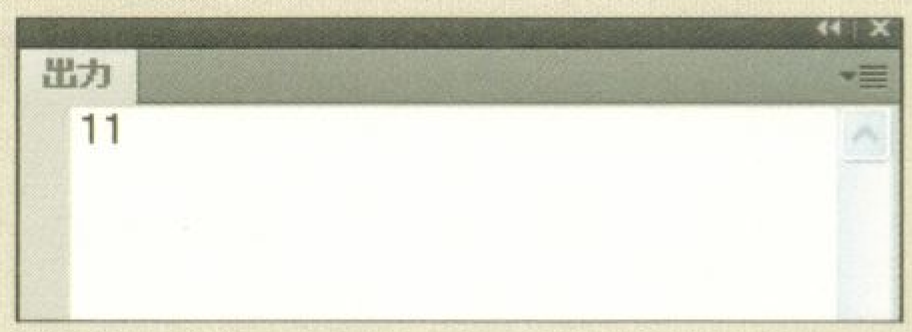
実行結果




```
var a = 1; ⇐ 変数aは数値
var b = "1"; ⇐ 変数bは文字列

trace(a+b);
```

実行結果

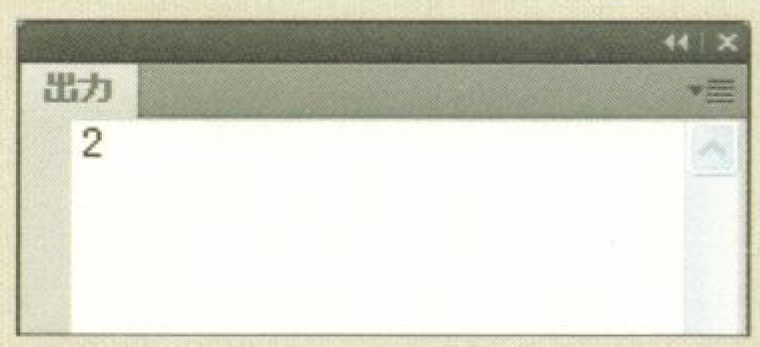


▶▶ 文字列と変数を足す

```
var a = 1; ⇐ 変数aに数値の1を代入
var b = a; ⇐ 変数bに変数a(つまり数値の1)を代入

trace(a+b);
```

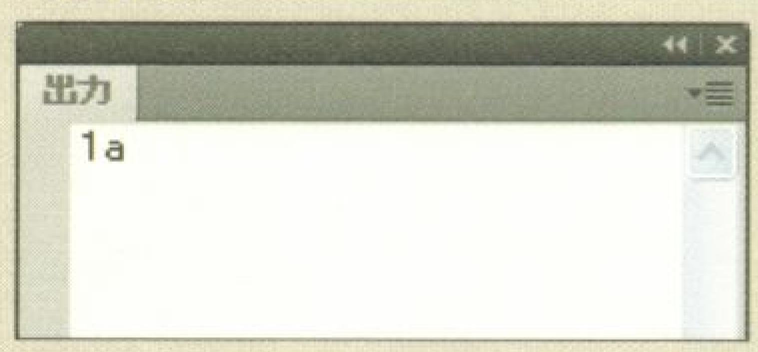
実行結果




```
var a = 1; ⇐ 変数aに数値の1を代入
var b = "a"; ⇐ 変数bに文字列aを代入

trace(a+b);
```

実行結果



このサンプルではわざと型指定を行っていない状態でスクリプトを書いてみましたが、実行結果が理解できたでしょうか。
最後のサンプルは少しトリッキーだったかもしれませんが、要は自分が扱っている変数は文字列なのか数値なのか、またそれらの変数は足したり引いたりできるか、その結果はどういう値が予想されるか、ということを考えながらプログラミングを行うとトラブルを少なくすることができます。

LECTURE 4-6

プログラミング［基本］

演算子

演算子とは、プログラミングで演算（数値や、変数を使って計算すること）を行うときに使用する記号のことです。見慣れたものだと、「＋」「－」「×」「÷」があります。それでは、プログラミングの演算に役立つ演算子を紹介します。

合わせて読むと便利な項目

■ 変数 ………………………… P094へ

■ if ………………………… P107へ

■ trace ………………………… P071へ

■ for ………………………… P112へ

算術演算子

四則演算（足し算、引き算、掛け算、割り算）に用いる記号を紹介します。
またここでは、演算の対象となる値や変数のことをオペランドと言います。

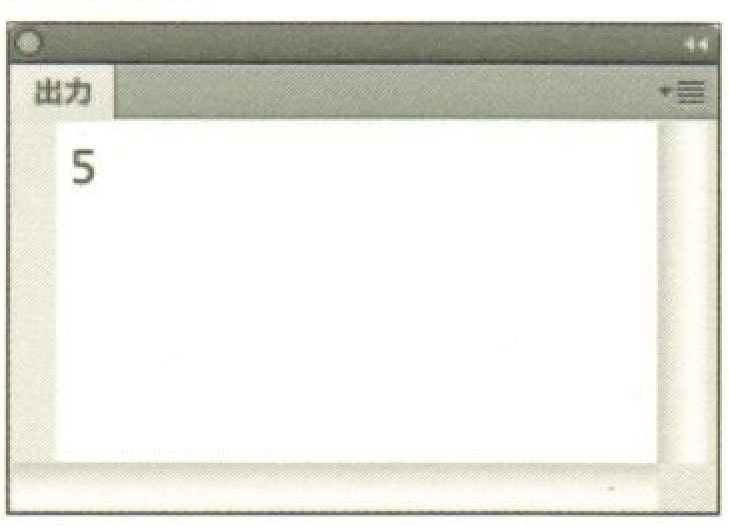
加算「＋」

この記号は2つの値を足すために用いられます。オペランドを加算するために、利用します。

数字の足し算

```
trace(3 + 2);
```

実行結果



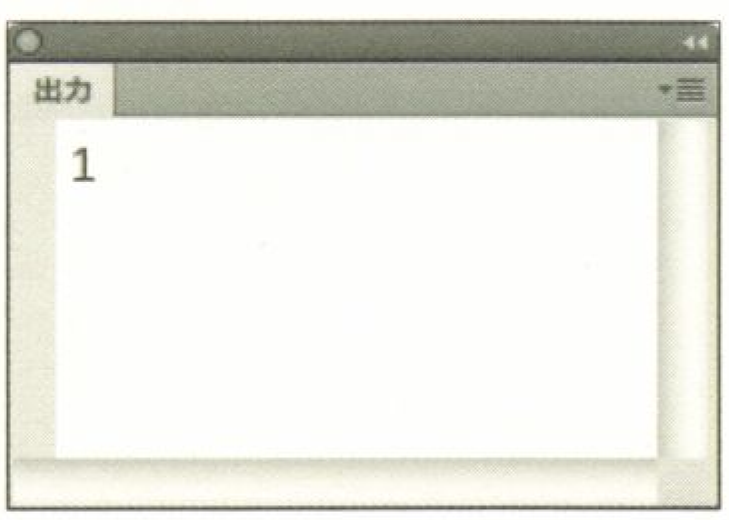
減算「－」

オペランドの差を出す際に用いられます。

数字の減算

```
trace(3 - 2);
```

実行結果



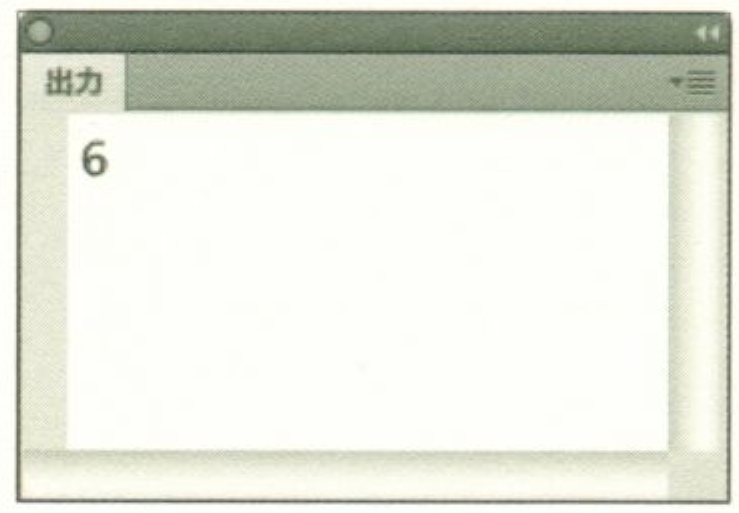
乗算「 * 」

2つのオペランドで掛け算（乗算）をするのに用いられます。
文字で書くときは、「×」ですが、ActionScriptでは、「*（アスタリスク）」が用いられます。

数字の掛け算(乗算)

```
trace(3 * 2);
```

実行結果



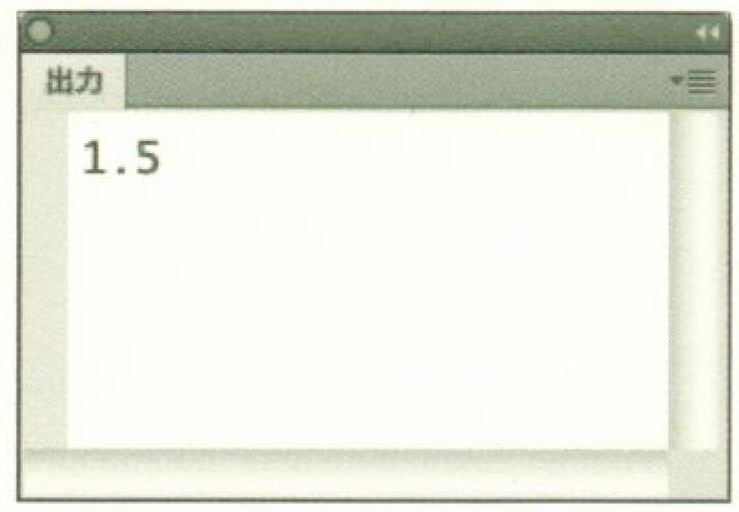
除算「 / 」

2つのオペランドで割り算（除算）をするのに用いられます。
文字で書くときは、「÷」ですが、ActionScriptでは、「/（スラッシュ）」が用いられます。

数字の割り算(除算)

```
trace(3 / 2);
```

実行結果



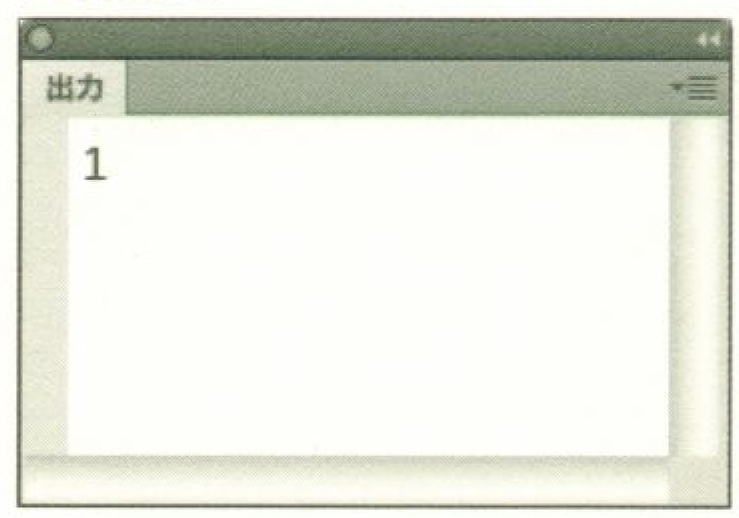
割ってあまりを出す「%」

前項で紹介した除算は、商を小数で求めるのに対し、除算したあまりを出すのがこの「%」です。

除算したあまり

```
trace(3 % 2);
```

実行結果



＋演算子による文字列の結合

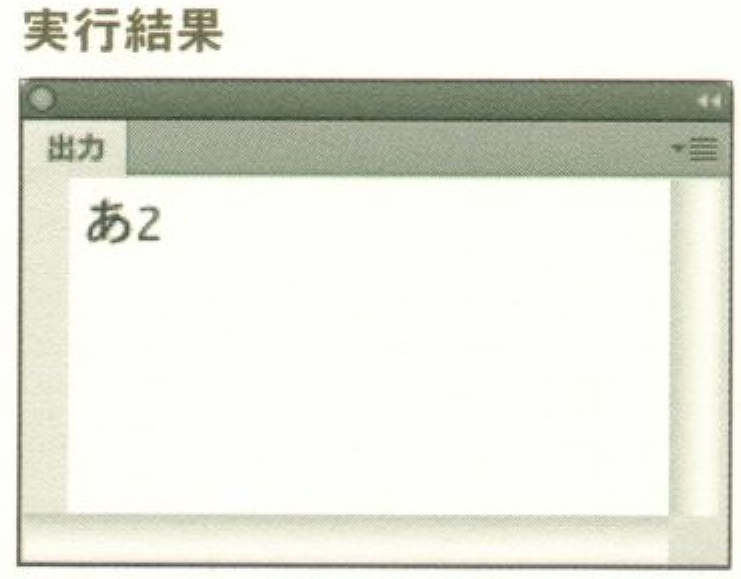
+演算子は文字列の結合にも用いられます。

＋演算子

文字列の足し算

```
trace("あ" + 2);
```

実行結果



複合代入演算子

名前はややこしいのですが、簡単にいうと変数を使って演算する際の計算式を省略する書き方です。
ここでは、「+=」「-=」「*=」「/=」の4つを紹介します。

「+=」

例えば、このように変数(a)に2を足した数を代入するとします。
これを、複合代入演算子で書くとこうなります。

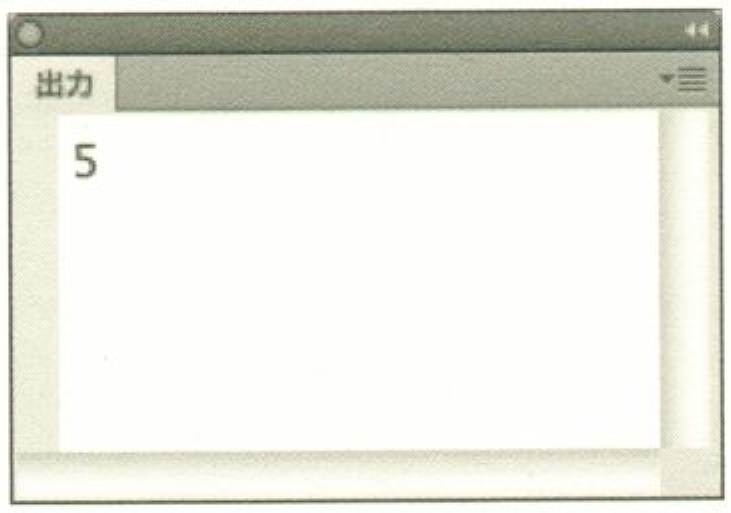
変数(a)に2を足す

```
//変数(a)に3を代入
var a:int =3;
//変数(a)に2を足した数を代入
a = a + 2;
//変数(a)をトレース
trace(a);
```

複合代入演算子を使用した場合

```
//変数(a)に3を代入
var a:int =3;
//変数(a)にaに2を足した数を代入
a += 2;
//変数(a)をトレース
trace(a);
```

実行結果



「-=」「*=」「/=」

減算

```
//変数(a)に3を代入
var a:int =3;
//変数(a)にaから2を引いた数を代入
//a = a - 2;と同じこと
a -= 2;
//変数(a)をトレース
trace(a); ⬅ 出力「1」
```

乗算

```
//変数(a)に3を代入
var a:int =3;
//変数(a)にaに2をかけた数を代入
//a = a * 2;と同じこと
a *= 2;
//変数(a)をトレース
trace(a); ⬅ 出力「6」
```

除算

```
//変数(a)に3を代入
var a:int = 3;
//変数(a)にaを2で割った数を代入
a /= 2;
//変数(a)をトレース
trace(a); ⬅ 出力「1.5」
```

プログラミングでは、ActionScriptを書いている際にエラーが頻繁に出ます。そのエラーの多くはヒューマンエラー(タイピングミスなど)にあります。いくらプログラミングが上達しても、このタイピングミスなどは起こるものです。そこで、そのミスが起きるプログラミングを書く量を下げる方法の1つとして、この「複合代入演算子」があります。

単項演算子

こちらも「複合代入演算子」と同じように省略型の演算子です。プログラミングでは、値を1つ増やしたり、1つ減らす計算が頻繁に行われます。例えば、フォトビューワーのサムネイルを1つずつ並べるときなど、forを使って繰り返し処理をする場合に用いられます。

■ for …………………………………………… P112へ

「++」

インクリメント演算子と呼ばれ、値を１増やします。

値を1増やす

```
//変数(a)に0を代入
var a:Number = 0;
//変数(a)に、aに1を足した数を代入
a++;
//変数(a)をトレース
trace(a);
```

実行結果



「−−」

デクリメント演算子と呼ばれ、値を１減らします。

値を1減らす

```
//変数(a)に10を代入
var a:Number = 10;
//変数(a)にaから1を引いた数を代入
a--;
//変数(a)をトレース
trace(a);
```

実行結果



この単項演算子ですが、「i++」のように変数の後ろに書くこともできるのですが、「++i」のように変数の前に書くこともできます。計算式の中に含めた場合、動作が変わってくるので、注意しましょう。

右の例のように、「++」を変数の後ろに書くと、代入後に加算されます。反対に「++」を変数の前に入れすると加算が先にされ、その後代入されます。必要な状況によって使い分けましょう。

```
//変数(a)に0を代入
var a:int = 0;
var b:int;
//変数(b)に、aを代入し、aにa+1を代入
b = a++;
//変数(b)をトレース
trace(b); ⬅[出力] パネルには「0」と出力
//変数(b)に、aに1を足した数を代入
b = ++a;
//変数(b)をトレース
trace(b); ⬅[出力] パネルには「2」と出力
```

比較演算子

比較演算子とは、値の大小を比較する条件式などで利用する演算子で、主にifで利用します。
ここでも紹介しますが、詳しくはifに書かれています。

■ if ………………………………………… P107へ

「<」左不等演算子

1番目のオペランドが2番目のオペランドよりも小さいかどうかをテストします。

「>」右不等演算子

1番目のオペランドが2番目のオペランドよりも大きいかどうかをテストします。

「<=」等価左不等演算子

1番目のオペランドが2番目のオペランドと同じ、もしくは2番目のオペランドより小さいかどうかをテストします。

「>=」等価右不等演算子

1番目のオペランドが2番目のオペランドと同じ、もしくは2番目のオペランドより大きいかどうかをテストします。

「==」

2つのオペランドが等しいかどうかをテストします。
数値を判定する場合にも使えますが、文字列を判定することもできます。

「!=」

2つのオペランドが等しくないかどうかをテストします。

論理演算子

主にifで利用される演算子で、比較演算子を使って比較をする際に、比較条件を複数指定する際に利用します。
こちらもifで詳しく取り扱っています。

■ if ………………………………………… P107へ

「&&」論理積

(条件A) && (条件B)と書き、条件Aと条件Bが共に真(true)の場合に、真(true)と判定されます。
例の場合、変数(a)が5未満で、変数(b)が10未満の場合に、"OK!"と出力します。

使用例

```
if(a < 5 && b < 10){
    trace("OK!");
}
```

「||」論理和

(条件A) || (条件B)と書き、条件Aもしくは、条件Bどちらか一方が真(true)の場合に、真(true)と判定されます。
例の場合、変数(a)が5未満、もしくは、変数(b)が10未満のどちらか一方の条件が真(true)の場合に、"OK!"と出力します。

使用例

```
if(a < 5 || b < 10){
    trace("OK!");
}
```

「！」

(!条件A)と書き、この条件以外である場合に真(true)と判定されます。NOTと呼び、この条件に合わない場合に真(true)とされます。

例1

```
var a = 10;

if(!(a < 5)){
    trace("OK!");  ⬅ aは5未満ではないので、「OK!」と出力
}
```

例2

```
var b = false;

if(!b){
    trace("偽です。");  ⬅ bは偽なので、「偽です。」と出力
}
```

条件演算子

条件演算子とは、真または偽の値を持つ論理式を元に、
1行で真(true)または、偽(false)の場合の挙動を指定できるものです。

書き方

(条件)?(評価がtrueの場合の返値):(評価がfalseの場合の返値)

例

```
//マウスが画面(500pxとする)上の半分より左にある場合は、左を、それ以外は右と出力する
function mousePos(ev:Event){
    (mouseX < 250) ? trace("マウスは画面左です") : trace("マウスは画面右です");
}

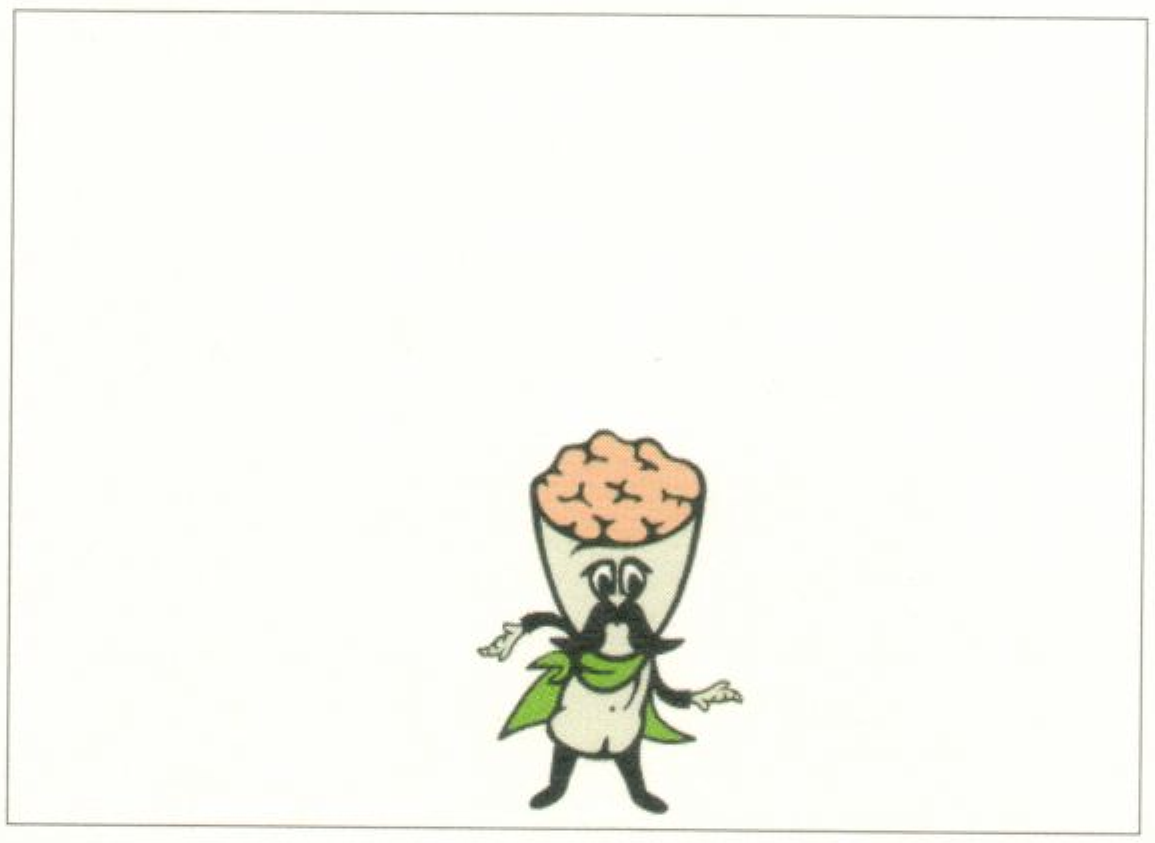
addEventListener(Event.ENTER_FRAME, mousePos);
```

画面の中心より右にマウスがある場合はキャラクターが右を、そうでない場合はキャラクターが左を向きます。

```
//繰り返し呼ばれるfunction
function mousePos(ev:Event){
    //マウスの位置がステージ(500px)の半分以下だったら左向く
    (mouseX < 250) ? chara.scaleX = -1 : chara.scaleX = 1;
}

//繰り返す関数を指定
addEventListener(Event.ENTER_FRAME, mousePos);
```

実行結果

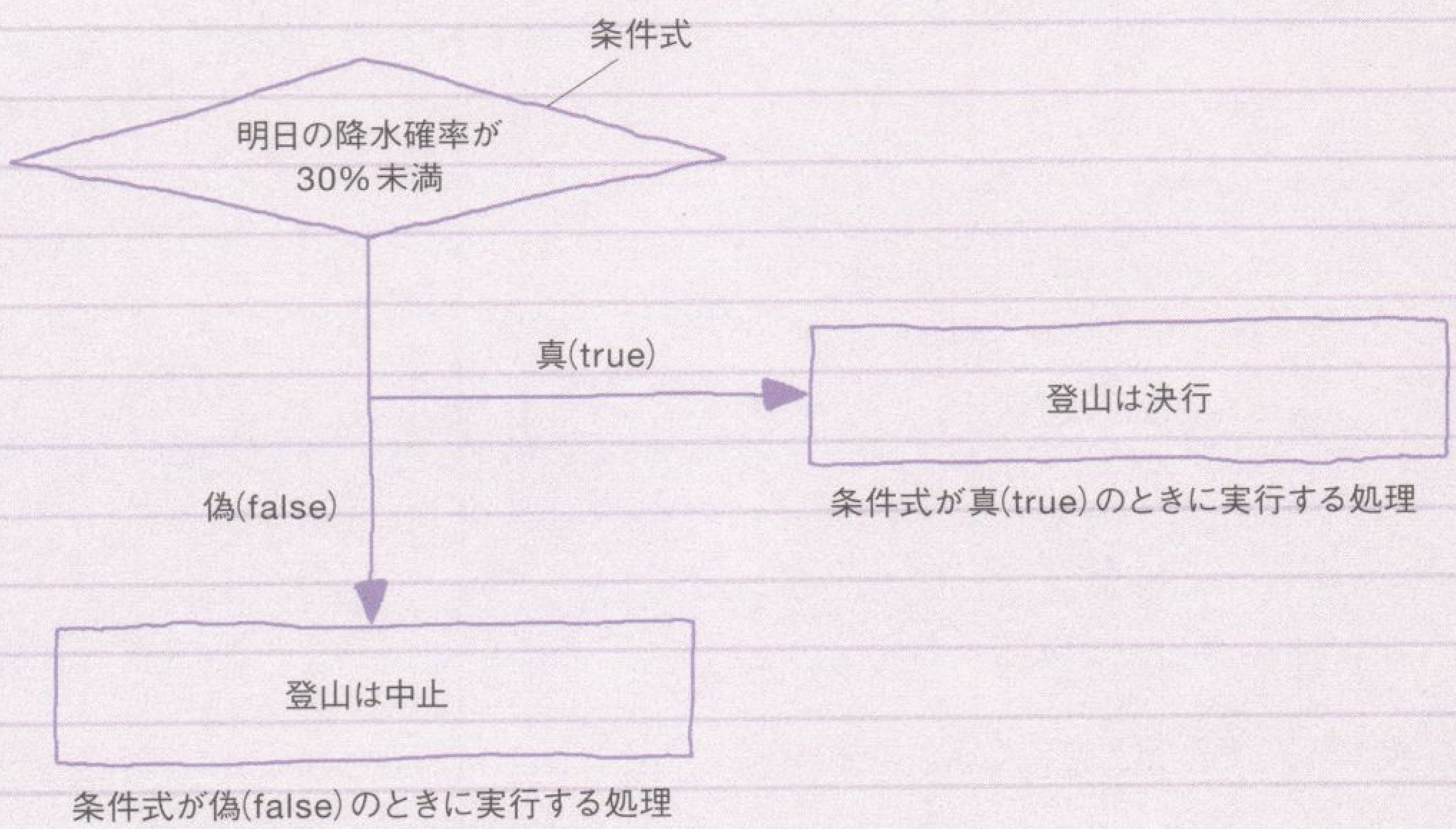


LECTURE 4-7

プログラミング［基本］

if

「明日の降水確率が30%未満なら登山は決行、30%以上なら登山は中止」というように、条件によってその後の処理を分岐させることを条件分岐と言います。if文はプログラム上で条件分岐を行う仕組みの1つで、ある条件に対して真（true）か偽（false）かによって、処理を分岐させることができます。



ifが使われているサンプル

■ PRACTICE 2：HTMLページを切り替える仕組みを作る ……………………… P199へ

合わせて読むと便利な項目

■ 変数 ……………………………… P094へ
■ 演算子 ……………………………… P101へ

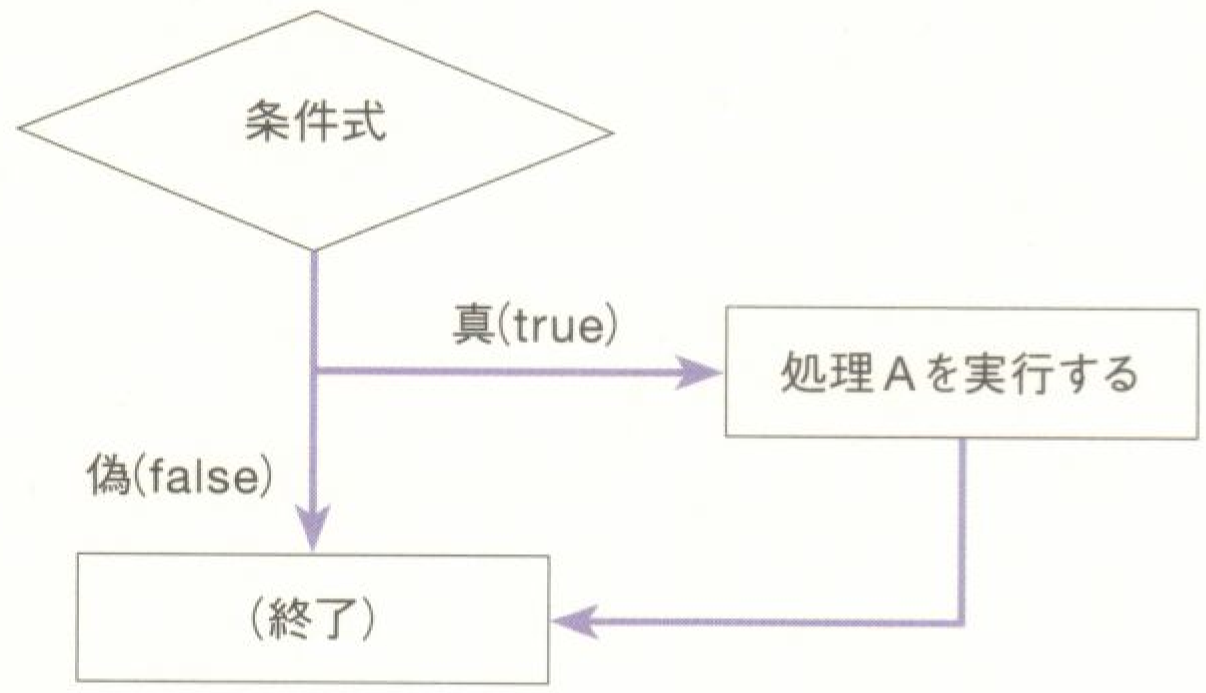
ifの基本的な使い方と演算子

それでは実際にActionScriptでifを使った条件分岐を見ていきます。
まずは基本的なif文の使い方と条件を作る際に必要な演算子について解説します。

▶▶ ifの基本構文

if文の基本的な書き方は次のようになります。
・条件が真なら処理を実行する（偽なら何もしない）

```
if (条件式) {
　条件式が真 (true) の時に実行する処理 (処理A)
}
```

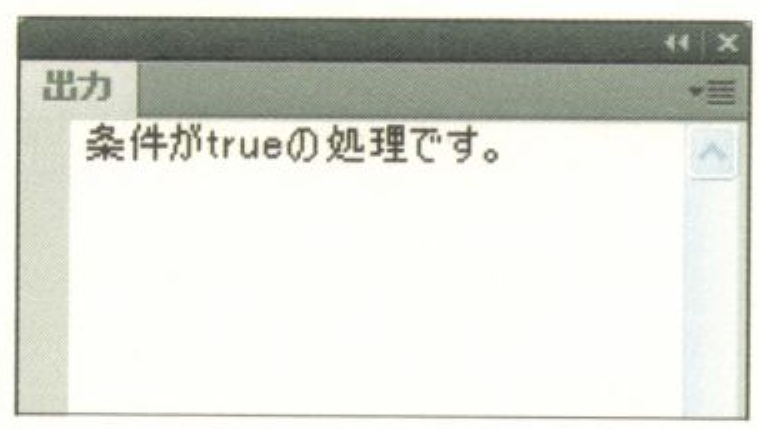


使用例

```
var num:Number = 100;

if(num == 100){  ⇐ 条件式「numが100ならば」
  trace("条件がtrueの処理です。");
  ⇑ 条件式がtrueのときに実行される処理
}
```

実行結果



▶▶ 比較演算子と論理演算子

条件式を書く際に必要となるのが、「○○が××と等しい」「○○は××以上」といった比較を表す比較演算子と、「○○と××がどちらとも真(true)」「○○が××ではない」といった論理値を扱う論理演算子です。

■ 演算子 ………………………………… P101へ

比較演算子

演算子	意味
==	等しい（等価）
!=	等しくない（不等価）
>	より大きい
<	より小さい
>=	以上
<=	以下
===	値もデータ型も等しい（厳密な等価）
!==	値もデータ型も等しくない（厳密な不等価）

論理演算子

演算子	意味
&&	論理積（AND） どちらもtrueのときにtrue。そうでなければfalse
||	論理和（OR） どちらか一方がtrueであればtrue そうでなければfalse
!	否定（NOT）

条件分岐のパターン（1）

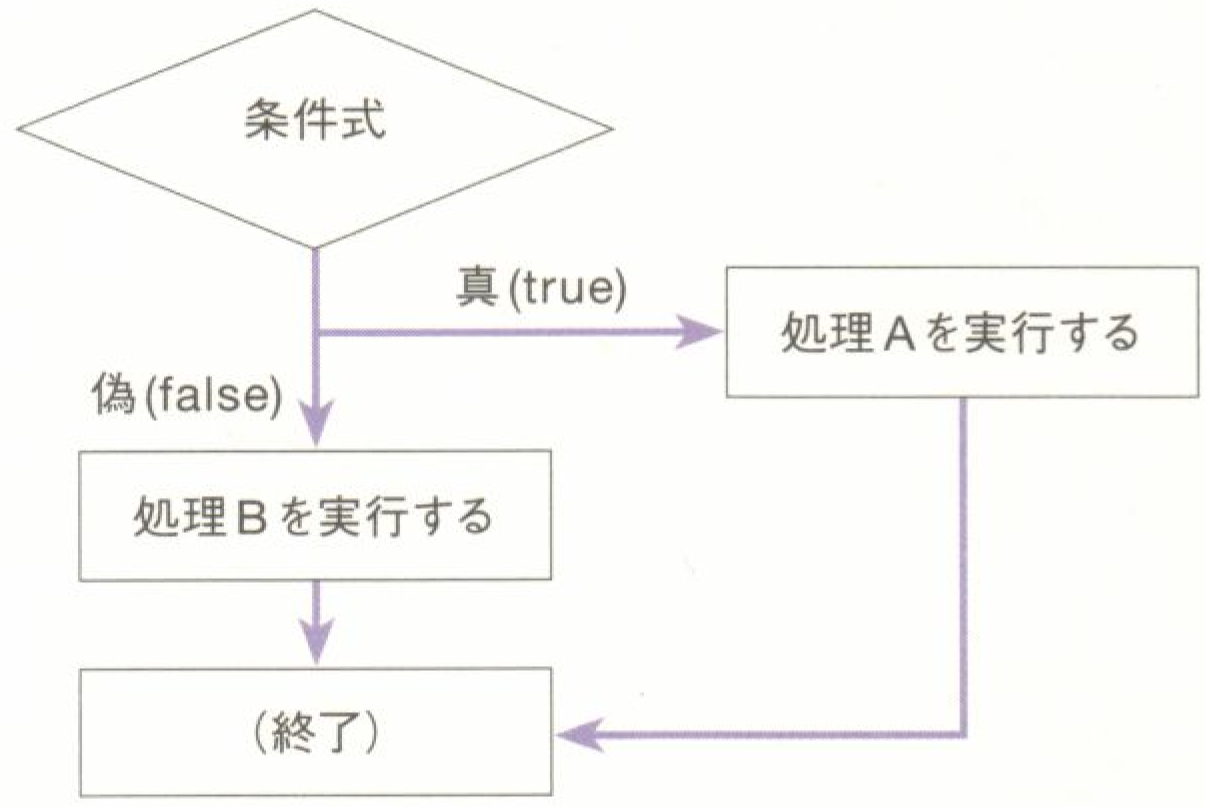
次に、条件分岐のパターンを見ていきます。
条件式が偽（false）の場合の処理を含めた書き方や、複数の条件式をまとめた場合などです。

if..else

条件式が偽（false）の場合の処理も加えたい場合は、if文にelseを追加します。

- 条件が真なら処理Aを実行する
- 偽なら処理Bを実行する

```
if（条件式）{
  条件式が真（true）の時に実行する処理（処理A）
} else {
  条件式が偽（false）の時に実行する処理（処理B）
}
```

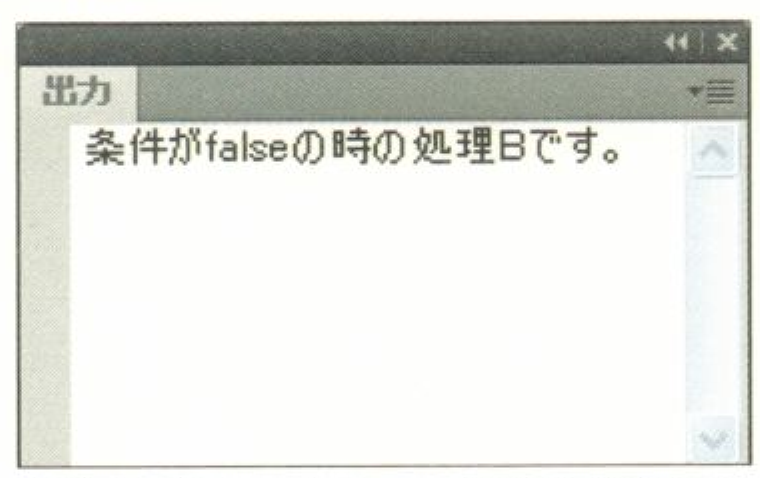


使用例

```
var num:Number = 100;

if(num == 50){   ← 条件式「numが50ならば」
  trace("条件がtrueのときの処理Aです。");
  ↑ 条件式がfalseなので、この処理は実行されない
} else {
  trace("条件がfalseのときの処理Bです。");
  ↑ 条件式がfalseなので、この処理が実行される
}
```

実行結果



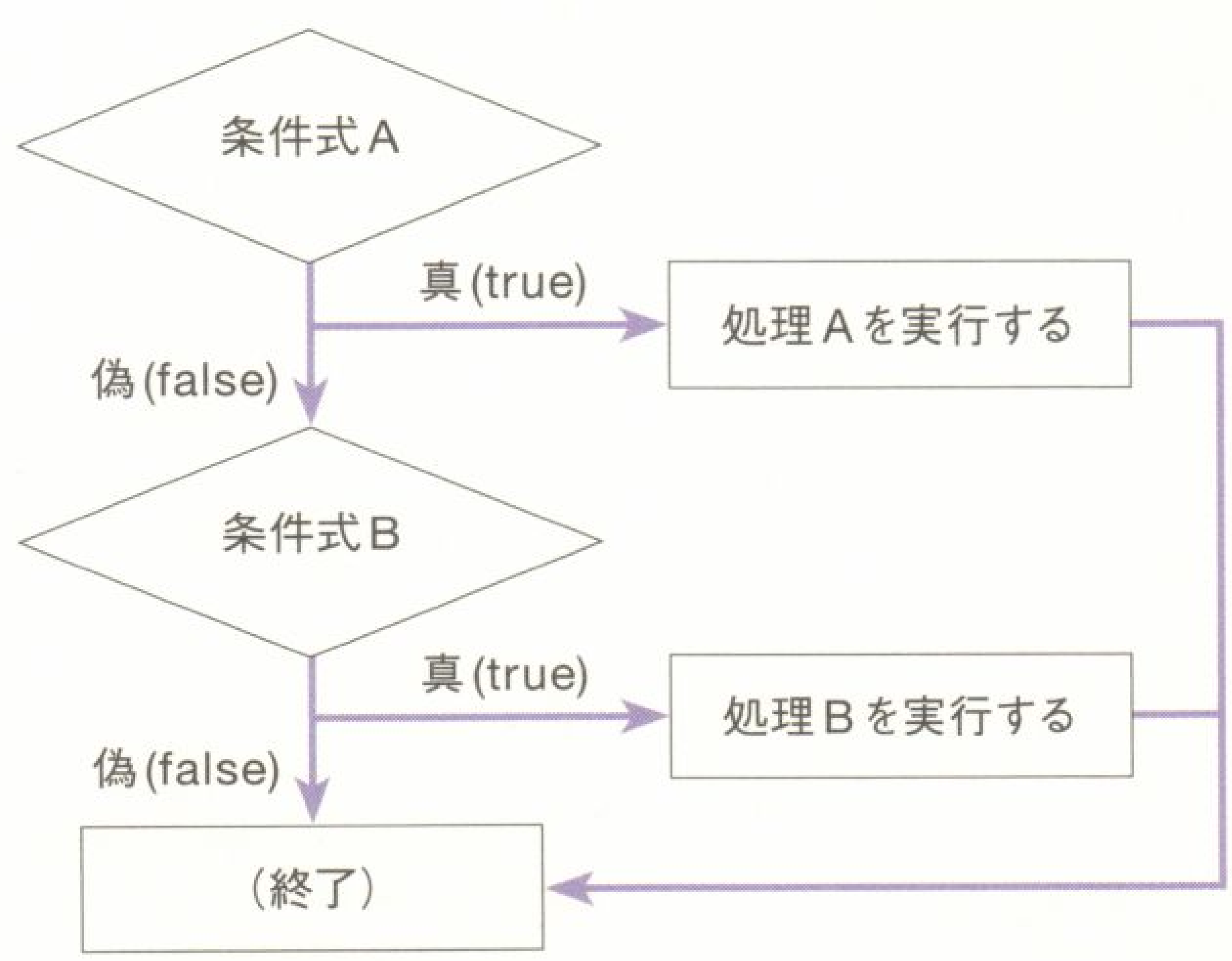
条件式がfalseなので、処理Aは実行されなかった

if..else if..

条件式が偽（false）のときには違う条件式をチェックする、という形で、複数の条件式をまとめた条件分岐を行う場合は、else ifを追加します。

- 条件Aが真なら処理Aを実行する
- 条件Aが偽なら→条件Bが真なら処理Bを実行する
- 条件Bが偽なら→処理Cを実行する

```
if（条件式A）{
  条件式Aが真（true）の時に実行する処理（処理A）
} else if（条件式B）{
  条件式Bが真（true）の時に実行する処理（処理B）
}
```

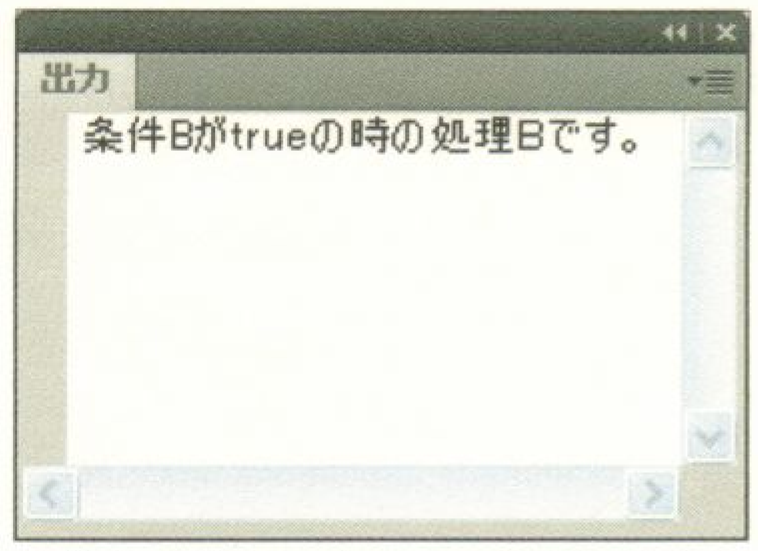


使用例

```
var num:Number = 100;

if(num == 50){ ⬅ 条件式A「numが50ならば」
   trace("条件Aがtrueのときの処理Aです。");
   ⬆ 条件式Aがfalseなので、この処理は実行されない
} else if(num > 50) {
   trace("条件Bがtrueのときの処理Bです。");
   ⬆ 条件式Bがtrueなので、この処理が実行される
} ⬅ 複数の条件式すべてがfalseのときの処理を加えたい
     場合、ここにelseを加える
```

実行結果



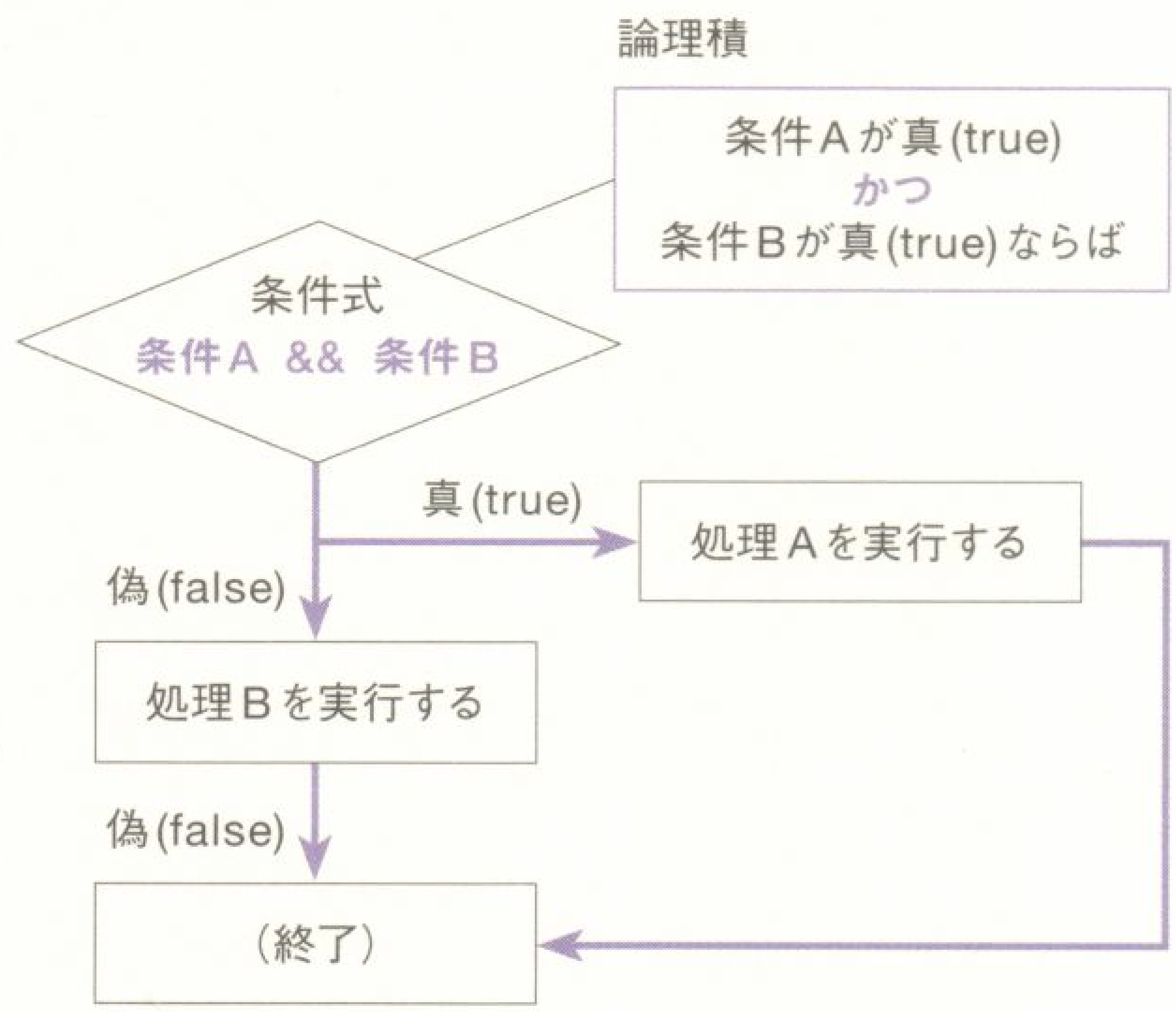
条件式Aがfalseなので、処理Aは実行されなかった

条件分岐のパターン(2)

最後に、論理演算子を使った複数の条件式の組み合わせのパターンを見ていきます。

▶▶ 論理演算子を使った条件式の組み合わせ

「条件Aが真(true)かつ条件Bが真(true)ならば処理を実行する」「条件Aまたは条件Bのいずれかがtrueなら処理を実行する」というように、1つの条件式の中に複数の条件を入れる形の条件分岐も、論理演算子を使って表現することが可能です。

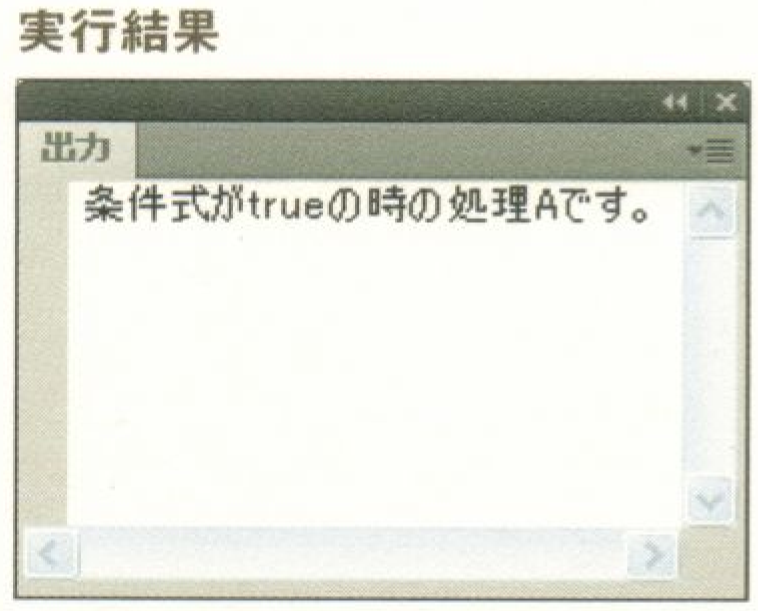


使用例

```
var num:Number = 100;

if(num == 100 && num >= 50){
   ⬆ 条件式「numが100かつnumが50以上ならば」
   trace("条件Aがtrueのときの処理Aです。");
   ⬆ 条件式がtrueなので、この処理が実行される
} else {
   trace("条件Bがtrueのときの処理Bです。");
   ⬆ 条件式がtrueなので、この処理は実行されない
}
```

実行結果

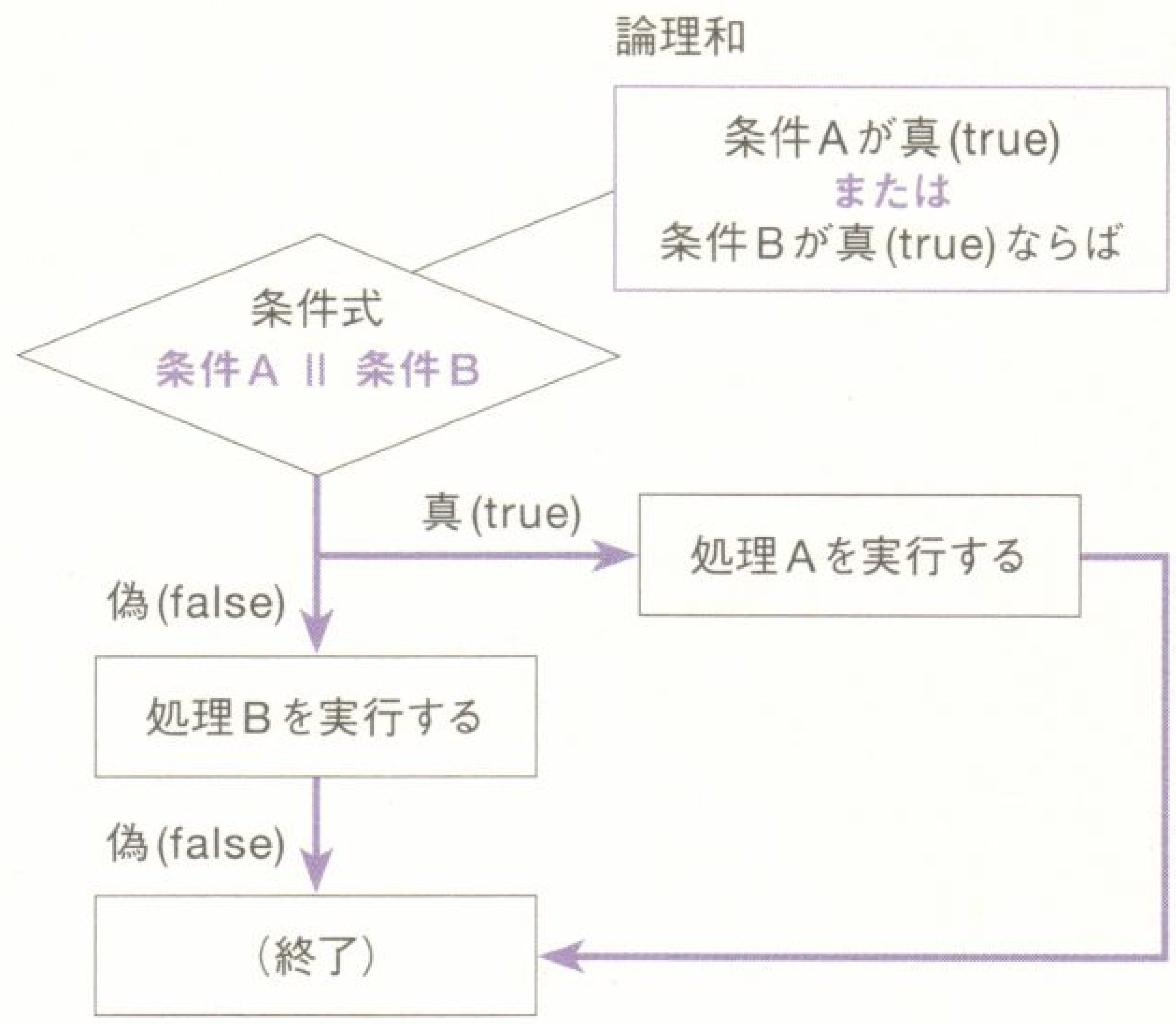


条件A、条件Bがいずれもtrueなので、処理Aが実行された

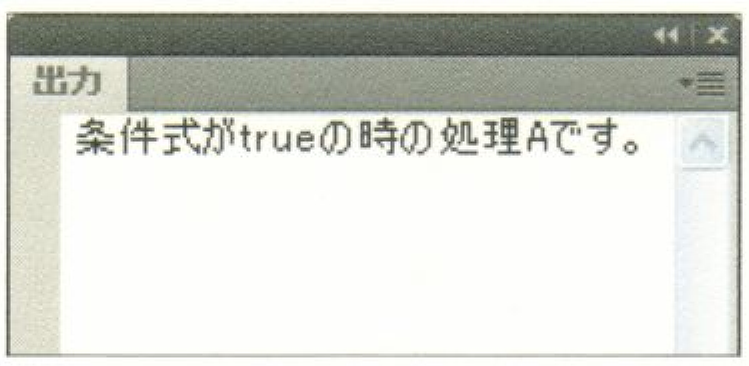
使用例

```
var num:Number = 100;

if(num != 100 || num > 50){
    ⬆条件式「numが100ではないか、またはnumが50
      より大きいならば」
    trace("条件式がtrueのときの処理Aです。");
    ⬆条件式がtrueなので、この処理が実行される
} else {
    trace("条件式がtrueのときの処理Bです。");
    ⬆条件式がtrueなので、この処理は実行されない
}
```



実行結果



条件Aはfalseだったが、条件Bがtrueなので、処理Aが実行された

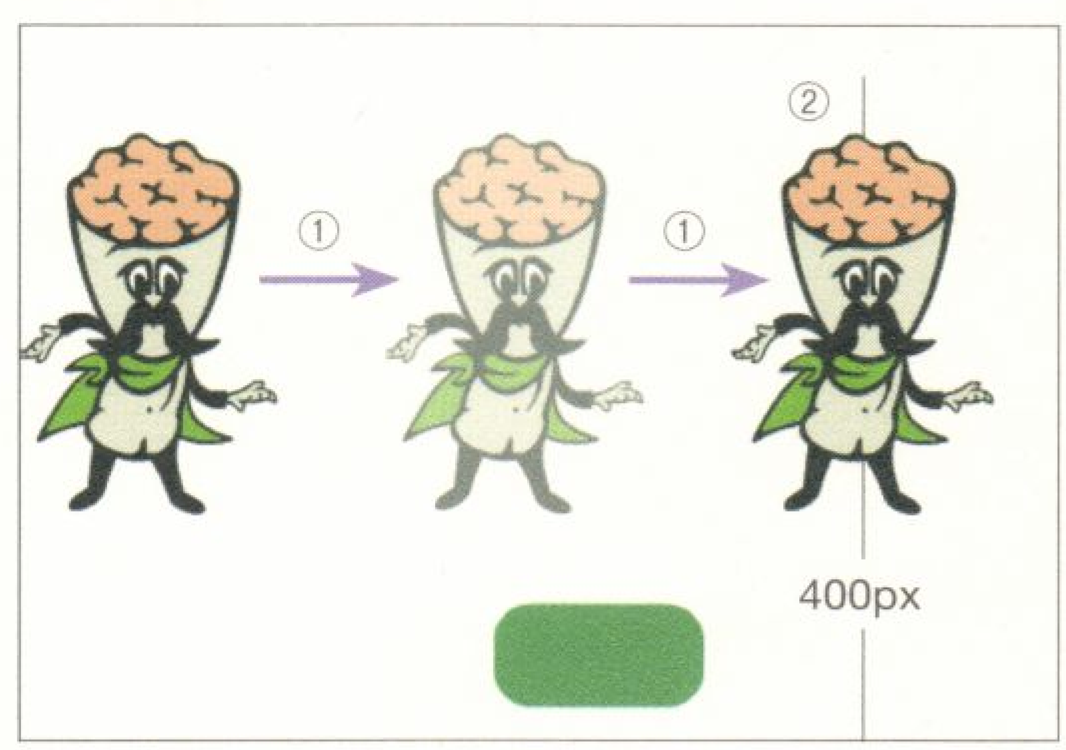
▶▶ サンプル：if文を使ってキャラクターの位置をコントロールする

if文を使って、「ボタンを押す度にキャラクターの位置が横方向に移動し、ある一定の場所に来たらそれ以上動かない」というサンプルを作ってみます。まずはメインタイムラインの1フレーム目に、キャラクター、ボタンをステージ上に配置し、それぞれのインスタンス名を、「chara」、「btn_mc」としておきます。
1フレーム目に以下のスクリプトを記述します。実行して、下の図のように動作すれば成功です。

```
//リスナー関数を作る
function moveMisoda(event){
    //charaのx座標が400ピクセル以上だったら
    if(chara.x >= 400){
        //イベントを削除する
        removeEventListener(MouseEvent.CLICK, moveMisoda);
    //charaのx座標が400ピクセル未満なら
    } else {
        //charaのx座標を20ピクセルずつ移動させる
        chara.x += 20;
    }
}

//ボタンと関数を結び付ける
btn_mc.addEventListener(MouseEvent.CLICK, moveMisoda);
```

実行結果



①ボタンをクリックする度、キャラクターが20pxずつ移動する。②400pxになると、それ以上移動しない。

これまで見てきたように、ifを使った条件分岐はelse ifや論理演算子を組み合わせていくことでさまざまな分岐をすることができます。
しかし、プログラミングをする際に、思い付いた順番で条件分岐をしていくと、if文が長くなり、その結果条件分岐で矛盾が生じたり、条件のチェック回数が多くなるために処理に時間がかかるようになることがあります。このようなことを避けるために、条件分岐をする際には、紙にフローを書くなどして設計を行い、分岐を必要最小限、かつ矛盾をなくすようにすることが大切です。

LECTURE 4-8

プログラミング［基本］

for

forは、同じような処理を繰り返し行うときに利用されます。プログラムをしていると、似通ったスクリプトを何度も書くことがあります。例えば、Webサイトでプルダウンメニューを作るとき（①）や、画面上に星を散りばめる（②）など、一定の処理を繰り返し行う際に利用します。

一定の処理を繰り返す例

①一定間隔でメニューを表示する



②ランダムに画面にオブジェクトを貼り付ける



forが使われているサンプル

■ PRACTICE 5：
サムネイルを動的に配置する ……… P224へ

合わせて読むと便利な項目

■ 配列 …………………………………… P115へ

forの使い方

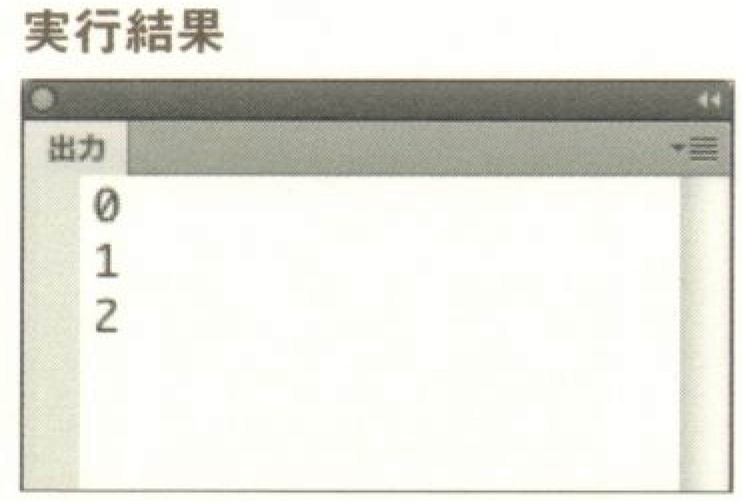
forの書き方と仕組みについて、解説していきましょう。

▶▶ forの書き方

まずは、書き方から見てみましょう。このスクリプトをメインタイムラインのフレームに書いて、ムービープレビューをすると、右図のように出力されます。これにより、このforに書かれた｛｝（中カッコ）の中が繰り返し動いていることが確認できます。

```
for(var i=0; i<3; i++){
   trace(i);
}
```

実行結果



forの仕組み

いったいどういう仕組みで動いているのか解説します。

```
        ①           ②                ④
for(初期値; 繰り返す条件; 次の処理の前に){
    処理内容
      ③
}
```

①初期値

forでは、始めに基準となる初期値を決めます。変数iに数値を入れて基準にし、この初期値を計算して回数を制御しています。

プログラムの多くは始めの数を0番とすることが多いため、初期値を0にしています。

ちなみに、配列も0番から数えます。

■ 配列 ………………………………… P115へ

②繰り返す条件

①で決めた初期値を使って、この条件を満たしていれば、③の繰り返し処理が行われます。

③処理内容

ここに繰り返したい処理を記載します。

②の条件をクリアしていれば、繰り返す処理を行います。

④次の処理の前に

③の処理が行われた後、ここに記載された処理が行われます。

つまり、forではまず、①を設定し、②で判定をし、条件がクリアした場合のみ③が実行され、④を行い、また②へ戻り、条件から外れるまで繰り返されます。

forの実行順番の仕組み

先ほどの出力結果を実際に動く順番を意識しながら、見ていきましょう。

①の初期値（i=0）から始まり、②⇨③⇨④⇨②⇨……という順で繰り返されます。

③で出力された結果では、「0、1、2」と3つですが、forの中では、iは3になってから停止しています。

```
         ①      ②     ④
for(var i=0; i<3; i++){
    trace( i )
       ③
}
```

	②繰り返す条件			③処理内容			④次の処理の前に
1回目	i = 0	⇨ OK	⇨	出力：0	⇨ 次	⇨	i = 1
2回目	i = 1	⇨ OK	⇨	出力：1	⇨ 次	⇨	i = 2
3回目	i = 2	⇨ OK	⇨	出力：2	⇨ 次	⇨	i = 3
4回目	i = 3	⇨ NG	⇨		⇨ 次	⇨	

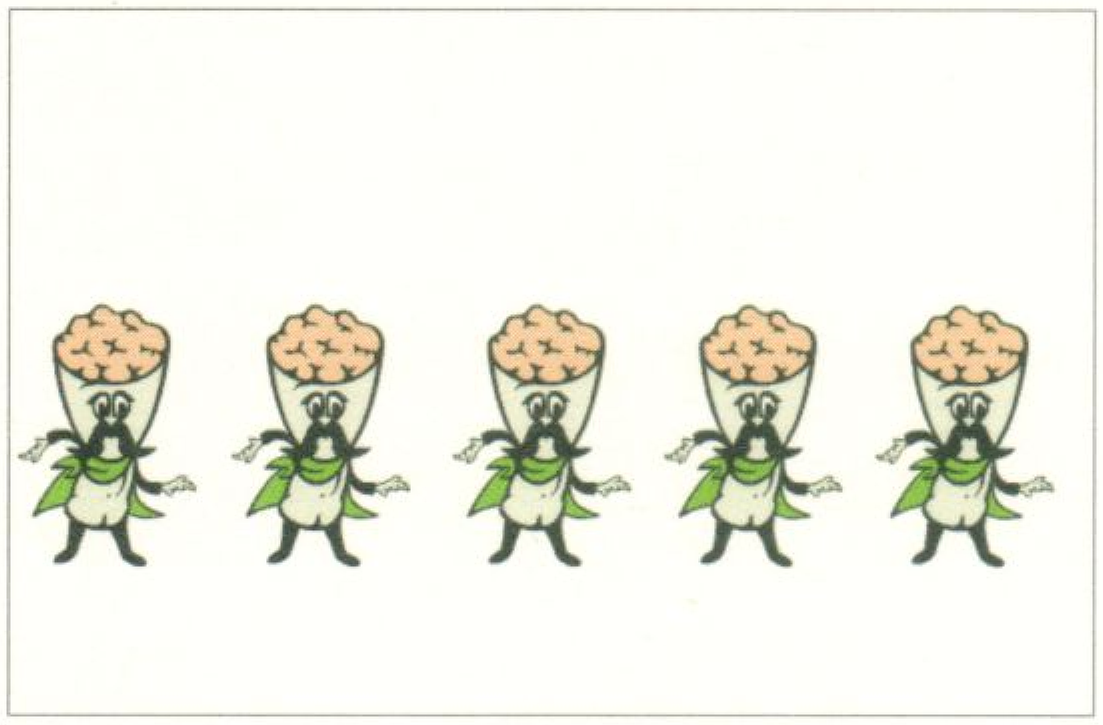
▶▶ サンプル：addChildや配列と組み合わせる

forは、配列と組み合わせると、威力を発揮します。キャラクターの配列分だけ、キャラクターをステージ上に表示する、というサンプルを紹介します。
最初に、forとキャラクター配置のみの例を見てみましょう。

addChildと組み合わせるサンプル

```
for (var i = 0; i < 5; i++) {
    // キャラクターを生成
    var mc : MovieClip = new Misoda();
    // ムービークリップを配置
    addChild ( mc ) ;
    mc.x = i * 150 + 100;
    ⬆150pxの間隔で横並びに配置(位置調整のため100pxを加算)
    mc.y = 200;  ⬅ y位置は200pxで統一
}
```

実行結果

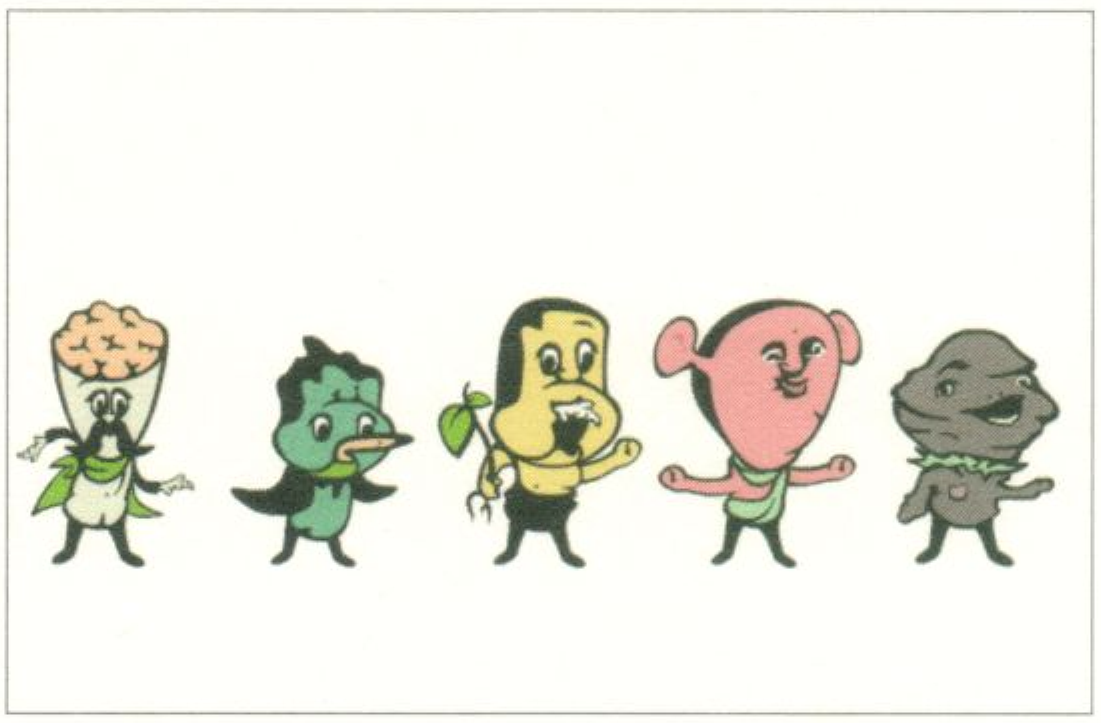


forで繰り返す回数分（ここではi=0〜i=4の5回）のキャラクター（Misoda）をステージ上に配置

次は、配列と組み合わせて、forで配列分の処理を繰り返すサンプルです。getDefinitionByNameについては、PRACTICE 6のEx.2（P255）を参照してください。

■配列 ………………………………………… P115へ
■addChild ………………………………………… P124へ

実行結果



最初に作成したキャラクターの配列分だけ、キャラクターをステージ上に配置

addChildと配列を組み合わせるサンプル

```
// 配列を作成
var arrChara : Array = [ "Misoda", "Kotori", "Zen", "Budpad", "Snovicke" ] ;
for (var i = 0; i < arrChara.length; i++) {
    // 配列から名前を取得
    var chara : String = arrChara[ i ] ;
    // 名前からクラスを作成
    var myClass : Class = Class(getDefinitionByName( chara ) ) ;
    // キャラクターを生成
    var mc : MovieClip = new myClass() ;
    // ムービークリップを配置
    addChild( mc ) ;
    mc.x = i * 150 + 100 ;
    mc.y = 200 ;
}
```

forはプログラムの効率アップになると言いましたが、それ以外にも利点があります。
それは、ミスの軽減です。プログラムのエラーの多くはヒューマンミステイク（タイピングミス）によるものです。そういったミスの多くは、単純なところにこそ発生します。forは無理に使う必要はありませんが、使うことによって単純な繰り返しの箇所を減らすことができるのも利用目的の1つです。

LECTURE 4-9

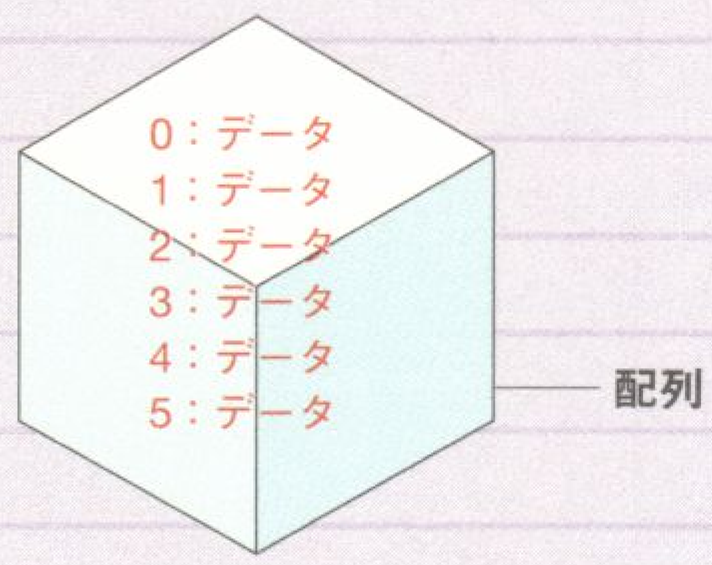
プログラミング［基本］

配列

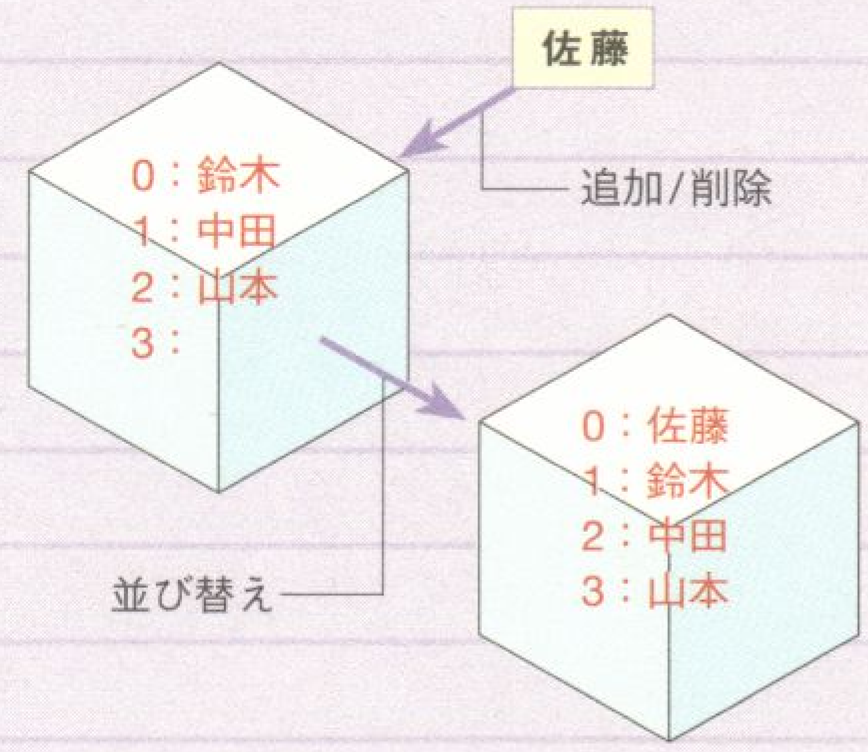
配列は複数のデータを保存、受け渡しするためのデータ形式です。配列のデータは番号を持っていて、順番を並び変えたり、特定のデータを追加、削除ができます。ただしこれはサーバにデータを残したり、ファイルに書き出したりするのとは別のもので、あくまで現在実行しているFlash内でのデータの保存です。

配列の特徴

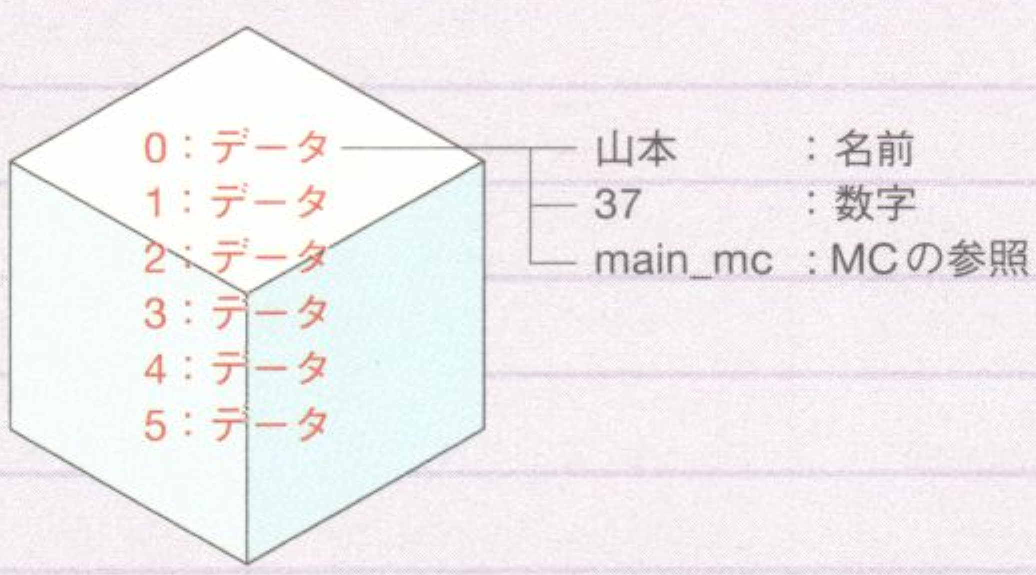
①配列は順番を持った箱



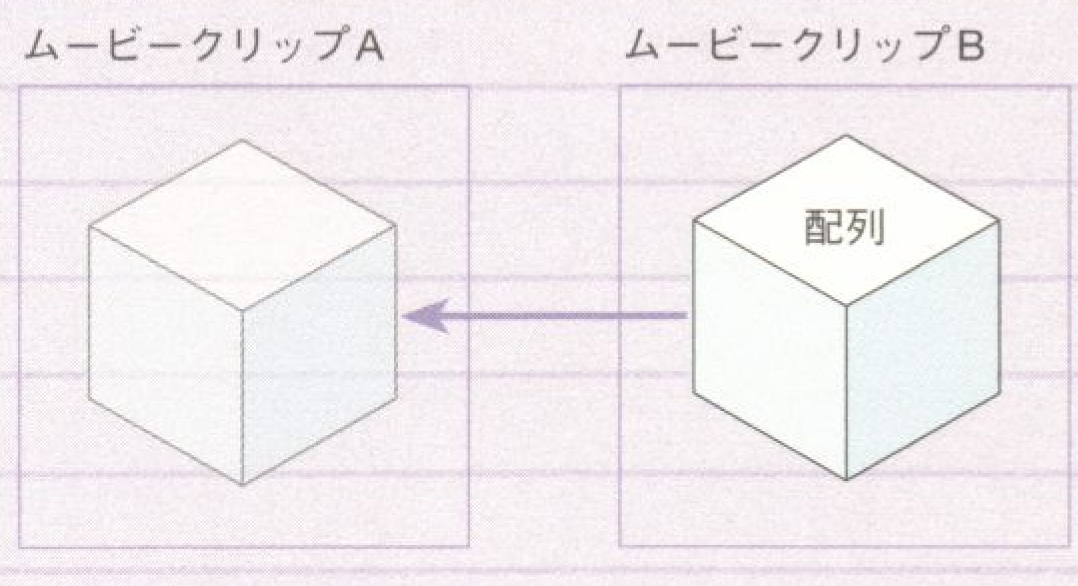
③順番を入れ替えたり、データの追加 / 削除ができる



②どんなものでも値として保存できる



④配列ごとデータの受け渡しができる



配列が使われているサンプル

■ PRACTICE 5：
サムネイルを動的に配置する ……… P224へ

合わせて読むと便利な項目

■ for …………………………………… P112へ
■ Object ………………………………… P128へ

配列の基本的な使い方

配列を使うにはArrayクラスを使用します。
まずはArrayクラスを使って配列に値を挿入、削除する方法を紹介します。

配列を初期化する

配列を使うにはまず変数を設定し、Arrayクラスを初期化（インスタンス化）する必要があります。初期化には2通りの方法があり、どちらも使っても同じ結果になります。

初期化方法①

```
var hobby:Array = new Array();
⬆ hobbyという変数で配列を初期化する
```

初期化方法②

```
var hobby:Array = [];
⬆ hobbyという変数で配列を初期化する
```

配列に値を入れる

配列に値を入れるには、基本は右の2つになります。どちらを使っても同じ結果です。

①あとから値を入れる方法

```
var hobby:Array = new Array();
⬆ hobbyという変数で配列を初期化します。
hobby[0] = "サッカー"; ⬅ 番号を付けて値を代入していく
hobby[1] = "酒";
hobby[2] = "Flash";
```

②最初から値を入れる方法

```
var hobby:Array = ["サッカー", "酒", "Flash"];
⬆ 初期化を値の代入を同時に行います。中身は方法①と同じ
```

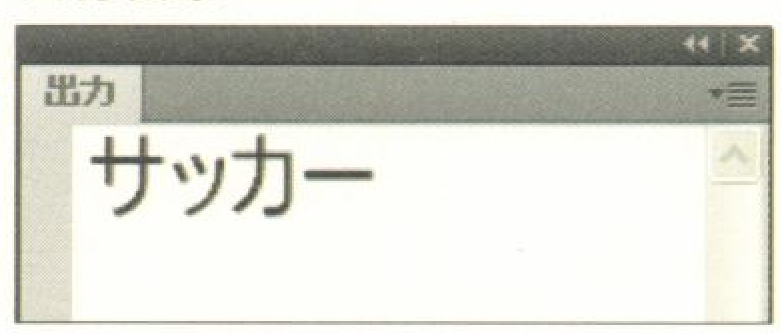
配列の値を確認する

配列の値は「変数名［番号］」で取り出すことができます。
配列の番号は0から始まりますので、1番目の値は「変数名［0］」になります。

使用例

```
var hobby:Array = new Array("サッカー", "酒", "Flash");
trace(hobby[0]); ⬅ 一番最初の値を取得する
```

実行結果



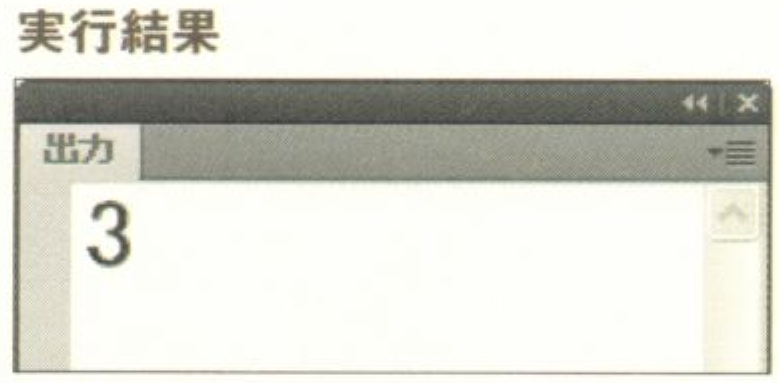
配列の長さを取得する

配列にどれだけの値が入っているかの長さが取得できます。

使用例

```
var hobby:Array = new Array("サッカー", "酒", "Flash");
trace(hobby.length); ⬅ 配列の要素の長さを取得する
```

実行結果



配列にデータを挿入、削除 (push,pop,shift,unshift)

配列にはデータを操作するメソッドが備わっており、それらを使うことで配列のデータを変更できます。
ここでは追加、削除のメソッドを紹介します。

push( )

配列の最後に値を追加します。

使い方

```
配列.push("値");
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.push("d");
trace(myArray);
```

実行結果

出力

```
a,b,c,d
```

pop( )

配列の最後の値を削除します。

使い方

```
配列.pop();
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.pop();
trace(myArray);
```

実行結果

出力

```
a,b
```

shift( )

配列の最初の値を削除します。

使い方

```
配列.shift();
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.shift();
trace(myArray);
```

実行結果

出力

```
b,c
```

unshift( )

配列の最初に値を追加します。

使い方

```
配列.unshift("値");
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.unshift("d");
trace(myArray);
```

実行結果

出力

```
d,a,b,c
```

配列のデータを並び替える、抜き取る (reverse,sort,splice)

配列のデータは並び変えや、指定した部分だけを抜き取ることができます。

reverse( )

配列の順番を逆にします。

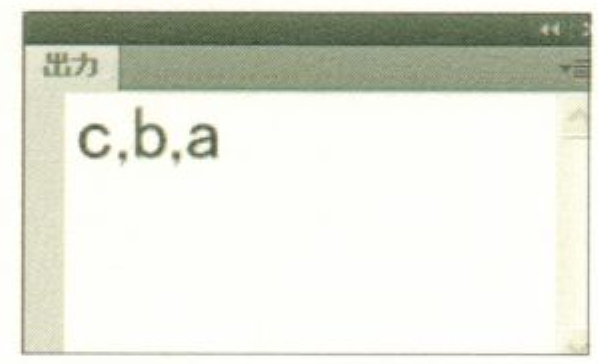
使い方

```
配列.reverse();
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.reverse();
trace(myArray);
```

実行結果



split( )

指定した値を複数抜き取ります。

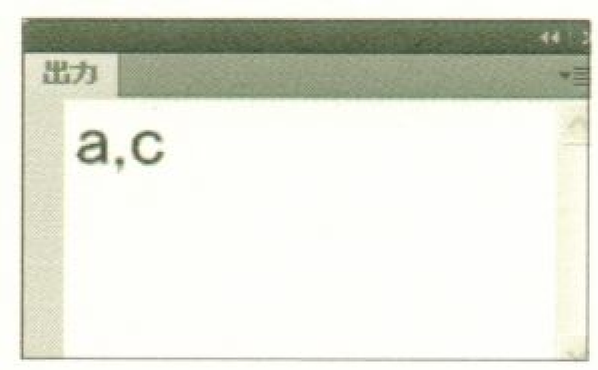
使い方

```
配列.splice(開始番号,抜き取る数の長さ);
```

使用例

```
var myArray:Array = ["a", "b", "c"];
myArray.splice(1, 1);
trace(myArray);
```

実行結果



sort( )

指定した並び替えの規則で配列の順番を入れ替えます。

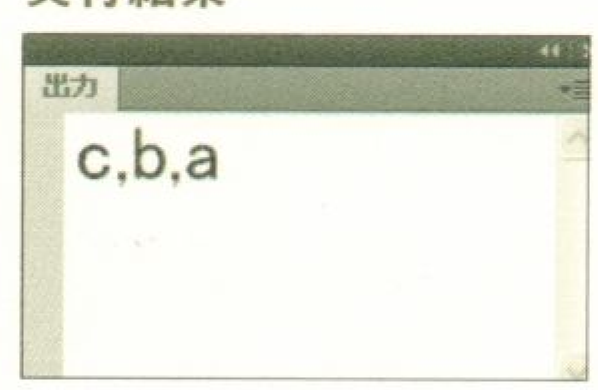
使い方

```
配列.sort(並び替えオプションを記述);
```

使用例

```
var myArray:Array = ["b", "a", "c"];
myArray.sort(Array.DESCENDING);
trace(myArray);
```

実行結果



並び替えオプション

Array.CASEINSENSITIVE	大文字と小文字を区別しないでソートします。
Array.DESCENDING	文字の大きい順でソートします。
Array.NUMERIC	値を数値として扱いソートします。
Array.RETURNINDEXEDARRAY	もとの配列を入れ替えることなくソートします。
Array.UNIQUESORT	同じ値が2つある場合はソートを終了します。

配列の組み合わせ技

配列の使い方の例として、ここではforやObjectとの組み合わせ方を紹介します。

配列とforの組み合わせ

配列には複数の要素が入っているので、それら要素を一度に取り出したり、追加したい場合があります。その場合はforと組み合わせることで、無駄なコードを書くことなく、配列を扱えます。

■ for ……………………………………………… P112へ

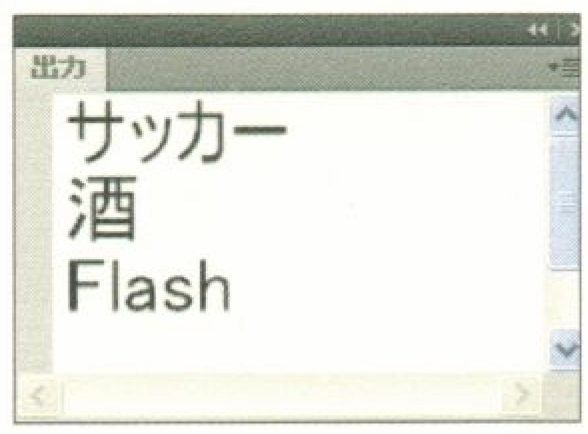
forで配列の値を取得する

for文で使用している変数「i」を使用することで、配列の中身を一度に取得できます。

使用例

```
var hobby:Array = ["サッカー", "酒", "Flash"];
for (var i:Number = 0; i<hobby.length; i++) {
⬆ 変数の長さ分ループする
   trace(hobby[i]);
   ⬆ 変数iを使うと、すべての配列要素にアクセスできる
}
```

実行結果



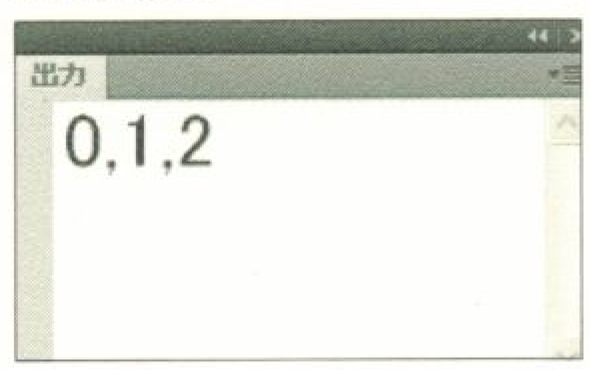
forで配列に値を追加する

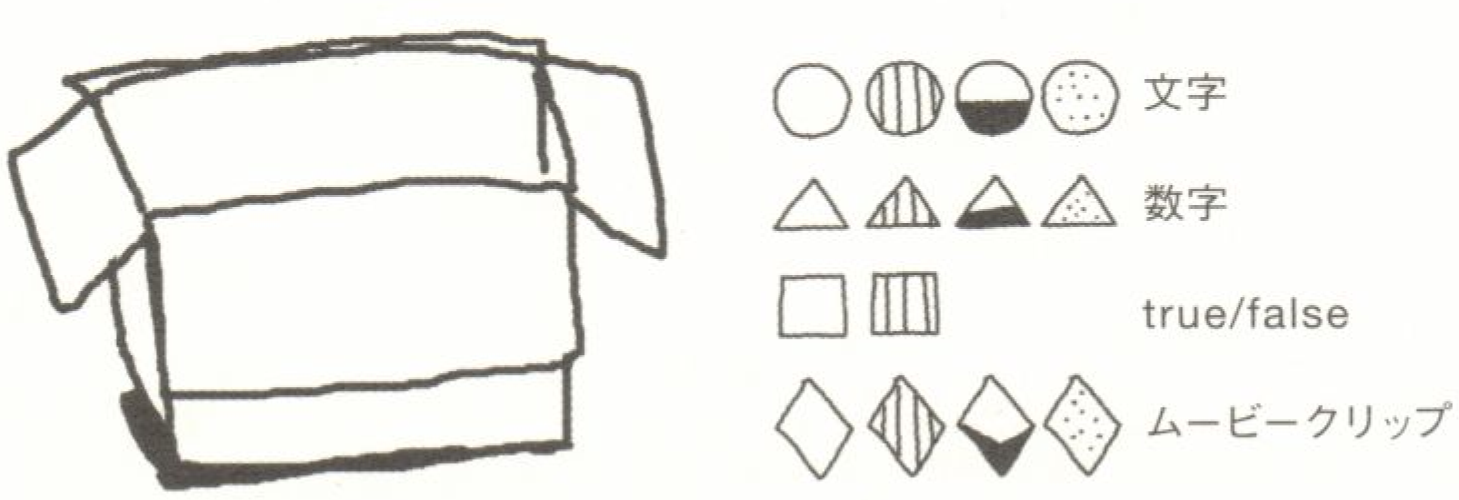
配列に値を追加するときはpushメソッドを使用します。

使用例

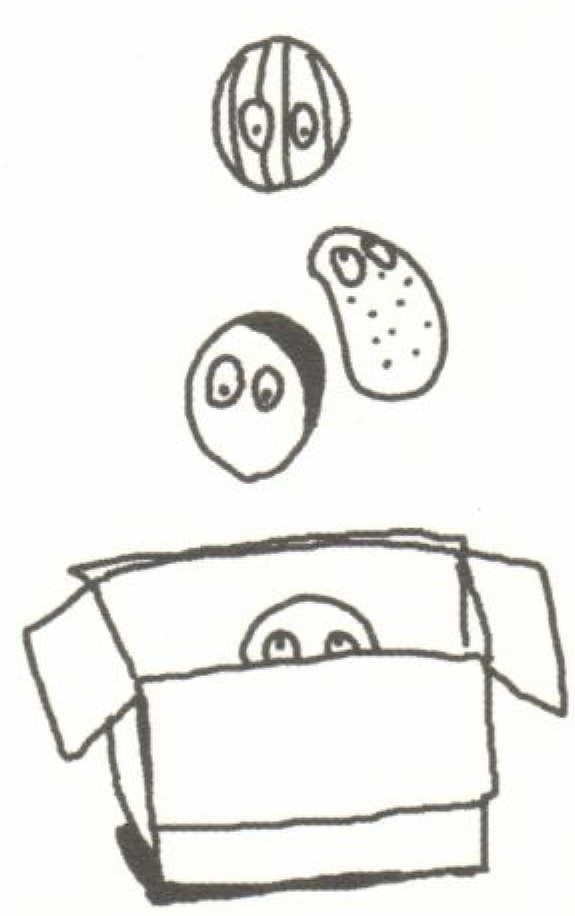
```
var hobby:Array = new Array();
for (var i:Number = 0; i < 3; i++) {
   hobby.push(i);  ⬅変数iを配列に追加
}
trace(hobby);
```

実行結果





①配列は順番を持ったデータの箱



②配列にはどんなものでも値として保存できる

▶▶ 配列とObjectの組み合わせ

配列には文字や数字だけでなく、オブジェクト型の値も入れることができます。オブジェクトには複数のデータを保存できるので、写真のURL、名前を1か所で管理できることが可能です。

配列に写真名とファイル名を保存する

写真の情報で写真名、ファイル名をの2つを管理したい場合は、まずはオブジェクトを作り、そこに2つのデータを保存します。

使用例

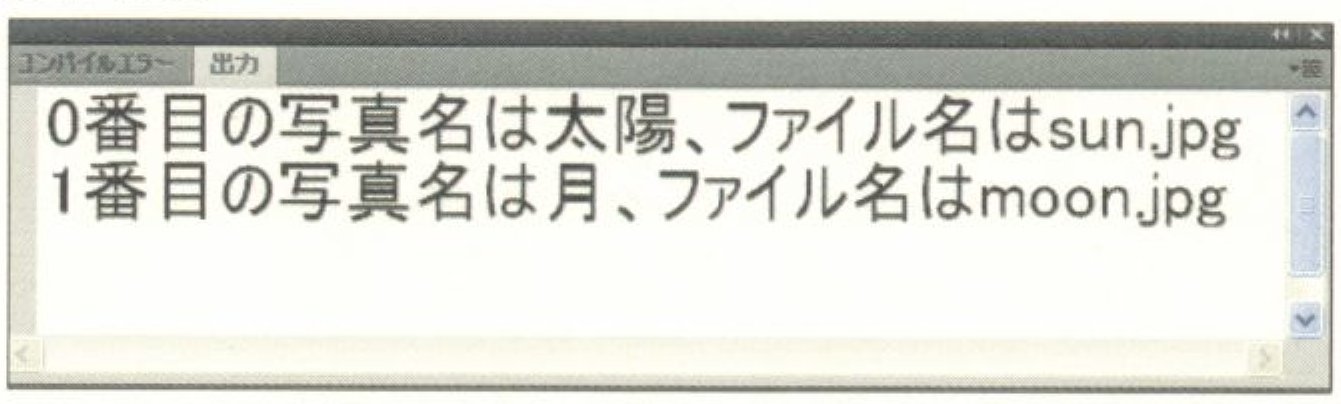
```
var pic0 = new Object();  ⬅1枚目の写真を設定
pic0.name = "太陽";
pic0.file = "sun.jpg";

var pic1 = new Object();  ⬅2枚目の写真を設定
pic1.name = "月";
pic1.file = "moon.jpg";

var picList:Array = new Array();  ⬅配列に写真情報を追加
picList.push(pic0);
picList.push(pic1);

for (var i:Number = 0; i < picList.length; i++) {
trace(i+"番目の写真名は" + picList[i].name + "、ファイル名は" + picList[i].file);
}
```

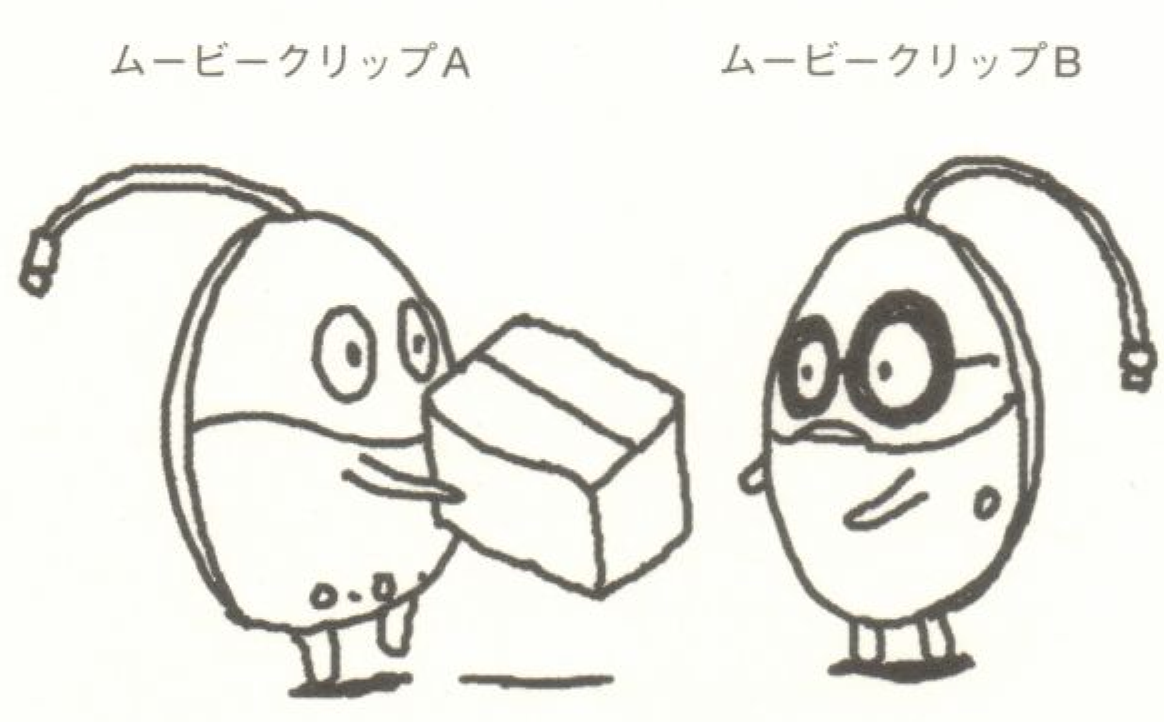
実行結果



■Object ······························ P128へ



③順番を入れ替えたり、データの追加 / 削除ができる



④配列ごとデータの受け渡しができる

LECTURE 4-10

プログラミング［基本］

ENTER_FRAME

ENTER_FRAMEは、ActionScriptでアニメーションを作る上で、なくてはならない機能です。このENTER_FRAMEを理解することで、マウスに追従する動きや、画像のフェードイン・フェードアウトなど、さまざまなビジュアルに関するプログラムを作ることができます。

ENTER_FRAMEが使われているサンプル

- PRACTICE 3：ENTER_FRAMEを利用する ……… P208へ

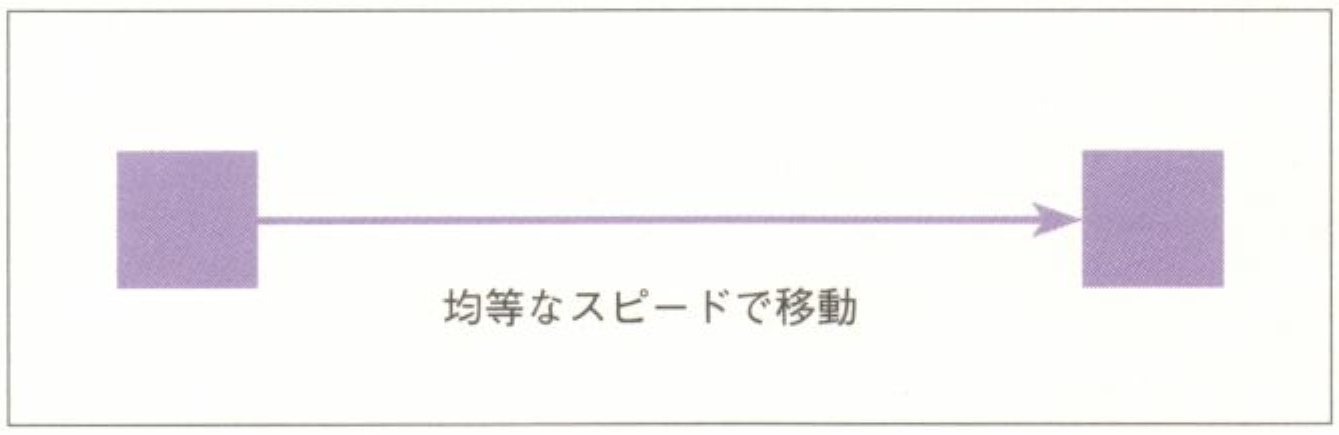
合わせて読むと便利な項目

- フレームレート …………………… P027へ
- function …………………………… P086へ
- addEventListener ………………… P090へ
- Timer ………………………………… P140へ

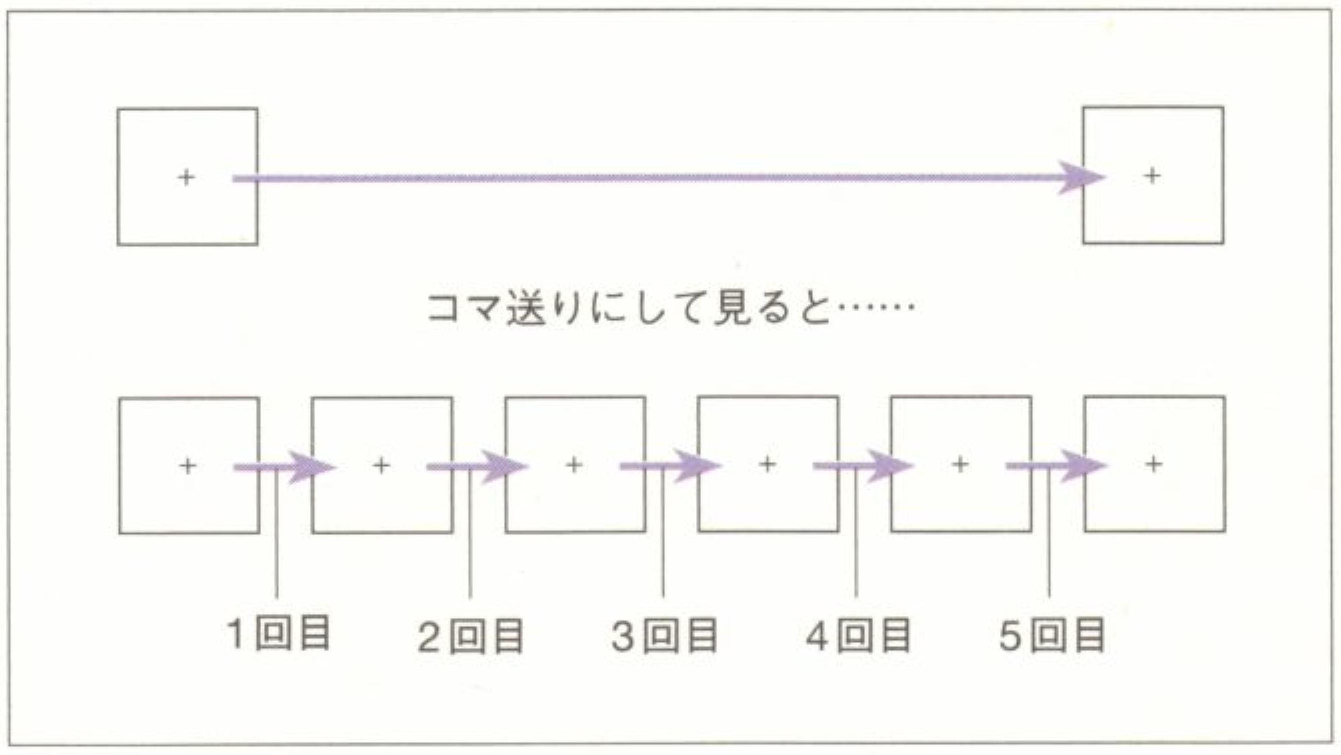
ENTER_FRAMEの考え方

ENTER_FRAMEは、プログラムを連続で実行したい場合に利用します。

右の図を見てみましょう。
これは四角い絵が左から右へ均等なスピードでアニメーションしているとイメージしてください。
左から右へはスムーズに流れているように見せますが、実際は右下図のように、コマ送りで表示がされているのです。



このようにアニメーションの一部分を切り出して、コマ送りにして見ると、1回ごとに右へ均等な距離を移動し、5回にわたって画面の右側へ移動しているということになります。
この一連の動作を見ることで、実は観えていない隙間の動きも人間の脳が勝手に作り出してこの間を動いたという認識になるのです。
ここで言う、1回ごとに動作している間隔を制御するのがフレームレートとなります。



フレームレートについて

ENTER_FRAMEを理解するにはまずは、フレームレートを理解しましょう。アニメーションの滑らかさに影響するフレームレートですが、ENTER_FRAMEにも影響を与えます。例えば、30fpsで実行されているFlashムービーの場合、1/30秒の間隔でプログラムが実行されていることになります。ActionScriptを利用したアニメーションでは、1秒間に30回も絵が更新されていることになります。

■ フレームレート…………………………………… P027へ

ENTER_FRAMEの使い方

ENTER_FRAMEの使い方を解説します。

▶▶ ENTER_FRAMEの使い方

まずは、繰り返し実行したいfunctionを用意します。
functionの引数には、Event型で設定しましょう。

```
//繰り返し呼ばれるfunction
function enterFrameHandler(ev:Event):void{
  //繰り返し呼ばれる処理
  trace("ENTER_FRAME実行中");
}
```

▶▶ ENTER_FRAMEを設定（開始）する

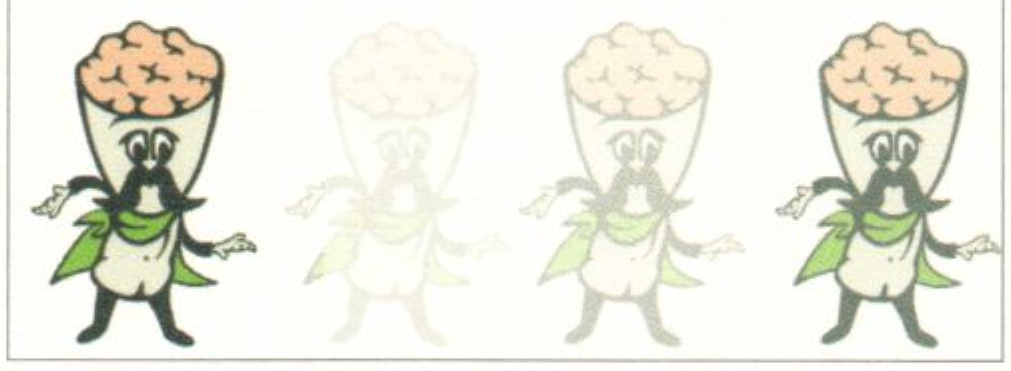
設定にはaddEventListenerを利用します。
上で用意した繰り返し実行したいfunctionを設定するとそのまま繰り返し処理が開始されます。

```
//繰り返す関数を指定
myTimer.addEventListener(Event.ENTER_FRAME,enterFrameHandler);
```

ステージ上に配置したムービークリップ(chara)のX位置を5pxずつ右へ移動する

```
//ムービークリップ(chara)のX位置を5pxずつ右へ移動する
function enterFrameHandler(ev:Event):void{
    //繰り返し呼ばれる処理
    chara.x = chara.x + 5;
}

//繰り返す関数を指定
addEventListener(Event.ENTER_FRAME, enterFrameHandler);
```



addEventListenerを使って、複数のfunctionを指定することで同時に実行することができます。

```
//繰り返す関数を指定
addEventListener(Event.ENTER_FRAME, enterFrameHandler);

//繰り返す関数2を指定
addEventListener(Event.ENTER_FRAME, enterFrameHandler2);

//繰り返す関数3を指定
addEventListener(Event.ENTER_FRAME, enterFrameHandler3);
```

ENTER_FRAMEを停止する

停止には、removeEventListenerを利用します。addEventListener同様に、このプログラムを実行すると、すぐに停止します。もう一度実行したい場合は、addEventListenerで再度設定しましょう。

```
//繰り返す関数を指定
myTimer.removeEventListener(Event.ENTER_FRAME,enterFrameHandler);
```

COLUMN

ENTER_FRAMEをTimer関数に置き換え

Flashで一定間隔で処理を行う関数は、ENTER_FRAMEのほかにもTimer関数が存在します。
同じように利用されるこの2つの処理ですが、大きく違うのは、ENTER_FRAMEは、繰り返す間隔がフレームレートに依存することです。最近ではWeb上のほとんどのコンテンツが24もしくは、30fpsで作られています。確かに滑らかには見えるのですが、ここまで細かく表示を更新すると却ってFlash Playerに描画の負担を与えてしまいます。結果、表示中にコマ落ちのように画面がスムーズに表示されなかったり、試合には止まってしまうこともあります。
一方、Timer関数は、フレームレートに依存することなく、くり返す間隔を調整できます。スムーズに表示を行う上でも、ENTER_FRAMEに替えて、このTimer関数が利用されることもあります。

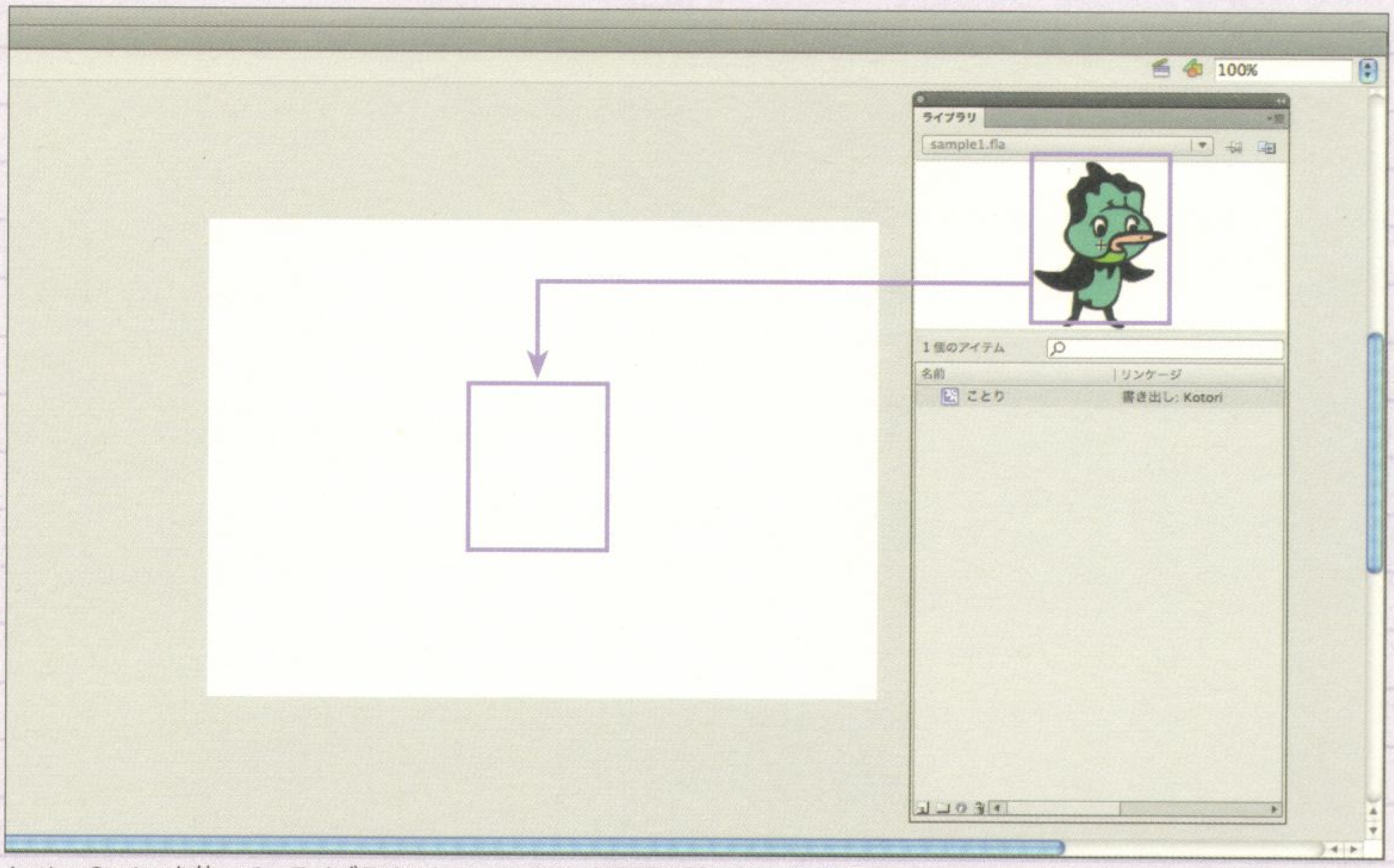
■ Timer関数 ······································· P140へ

LECTURE 4-11

プログラミング［基本］

addChild

addChildを利用すると、ライブラリにあるオブジェクトを画面上に貼り付けることができます。例えば、シューティングゲームの弾丸など、より動的なコンテンツ制作へ向いています。



ActionScriptを使って、ライブラリからMovieClipを画面に貼り付けます。

addChildが使われているサンプル

■ PRACTICE 3：
円のアニメーションを作る…………… P206へ

合わせて読むと便利な項目

■ Flashの画面構成（レイヤー部分）… P024へ
■ シンボル ……………………………… P034へ

COLUMN

貼り付け可能な種類

addChildを使って画面上に貼り付けることができる種類はムービークリップやボタンなどのシンボルだけでなく、ビットマップやビデオオブジェクトも貼り付けることが可能です。

貼り付け可能な種類
ムービークリップ
ボタン
ビットマップ
ビデオ
TextField

addChildの使い方

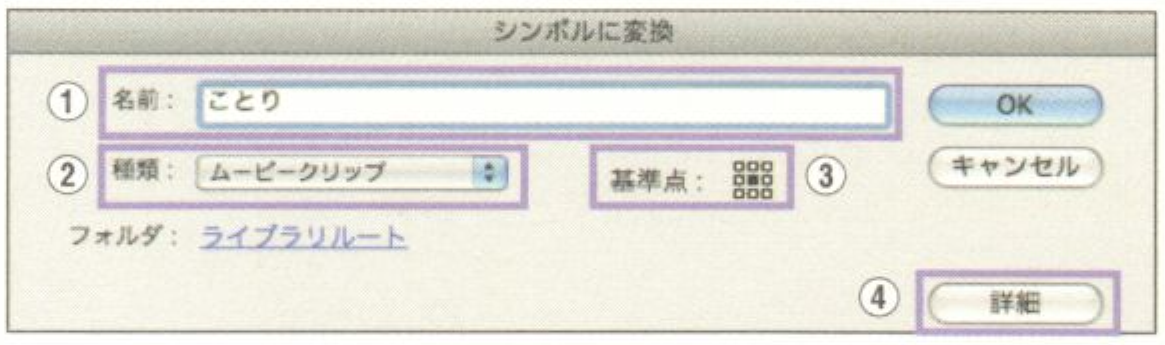
addChildでMovieClipを画面に登場させ、removeChildで画面から消す、というのが基本的な使用法になります。手順を追って見てみましょう。

▶▶ ムービークリップを作成する

1. 配置に必要なシンボルを作りましょう。
まず、何かしら絵を描き、［修正→シンボルに変換］を選択します。

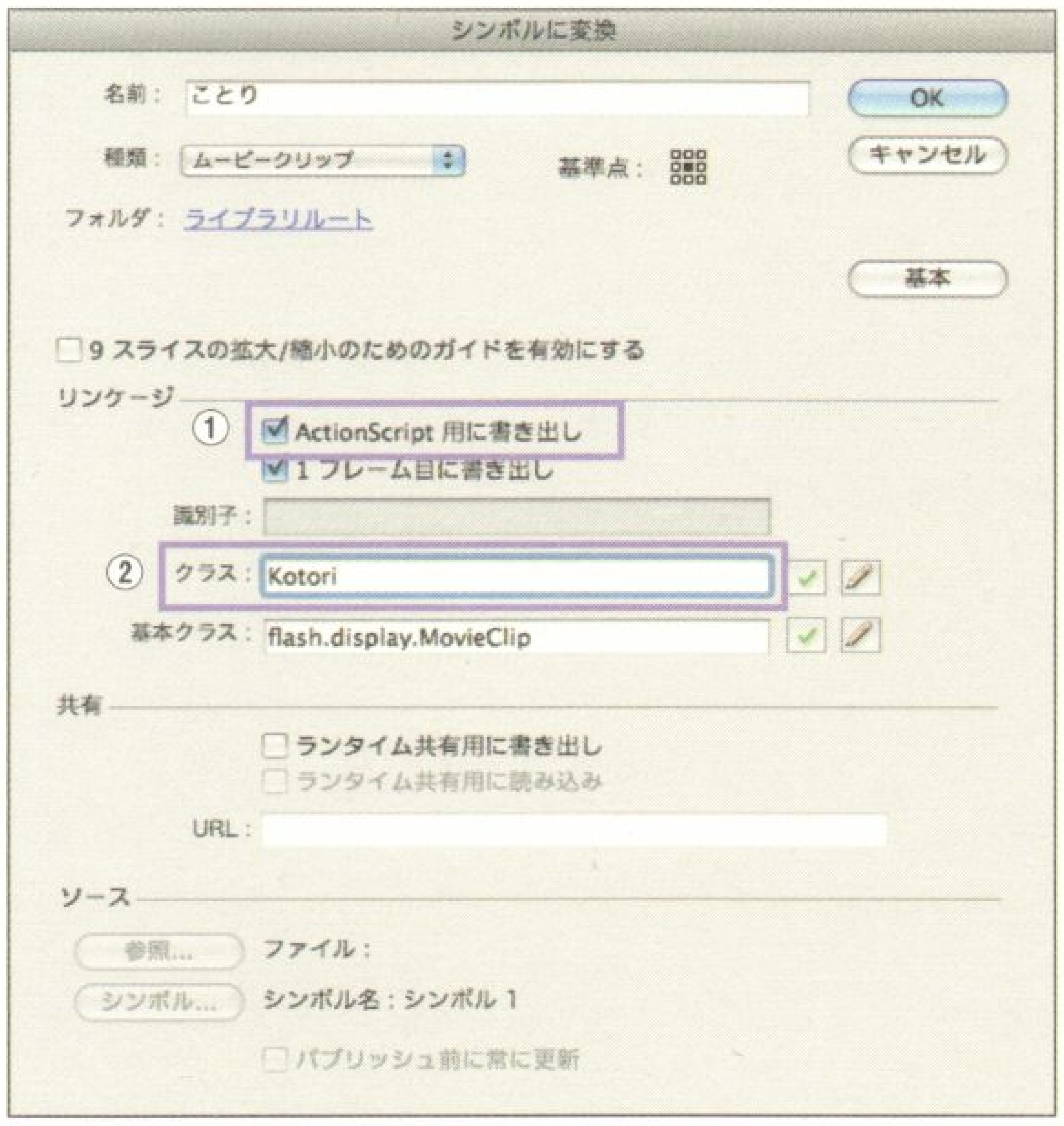


2. ムービークリップを作ります。
名前を付け(①)、種類をムービークリップ(②)にして、基準点の位置を中心にします(③)。
次に、詳細(④)をクリックします。



3. クラスを設定します。
「ActionScript用に書き出し」(①)にチェックを入れ、クラス(②)に、「Kotori」と入れます。
詳しくは、「3-5 クラス」の項を確認してください。

■ クラス …………………………………… P074へ

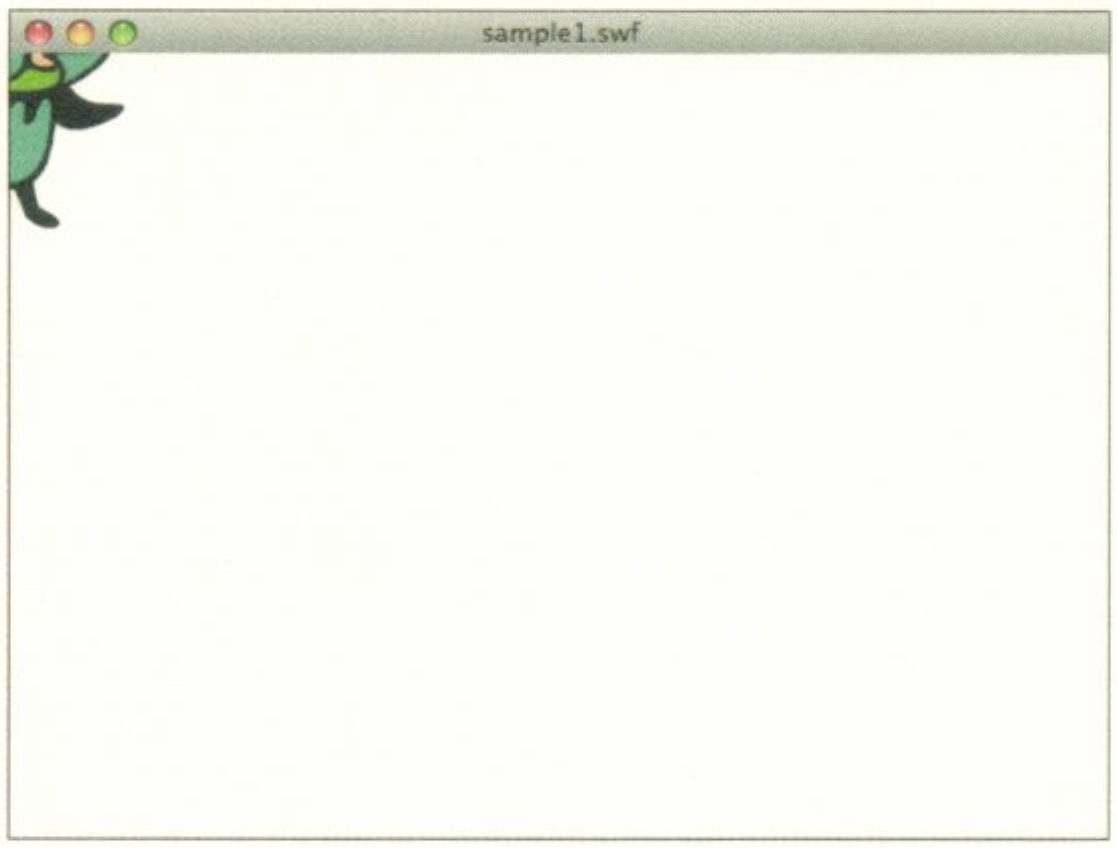


▶▶ ムービークリップを画面に配置する

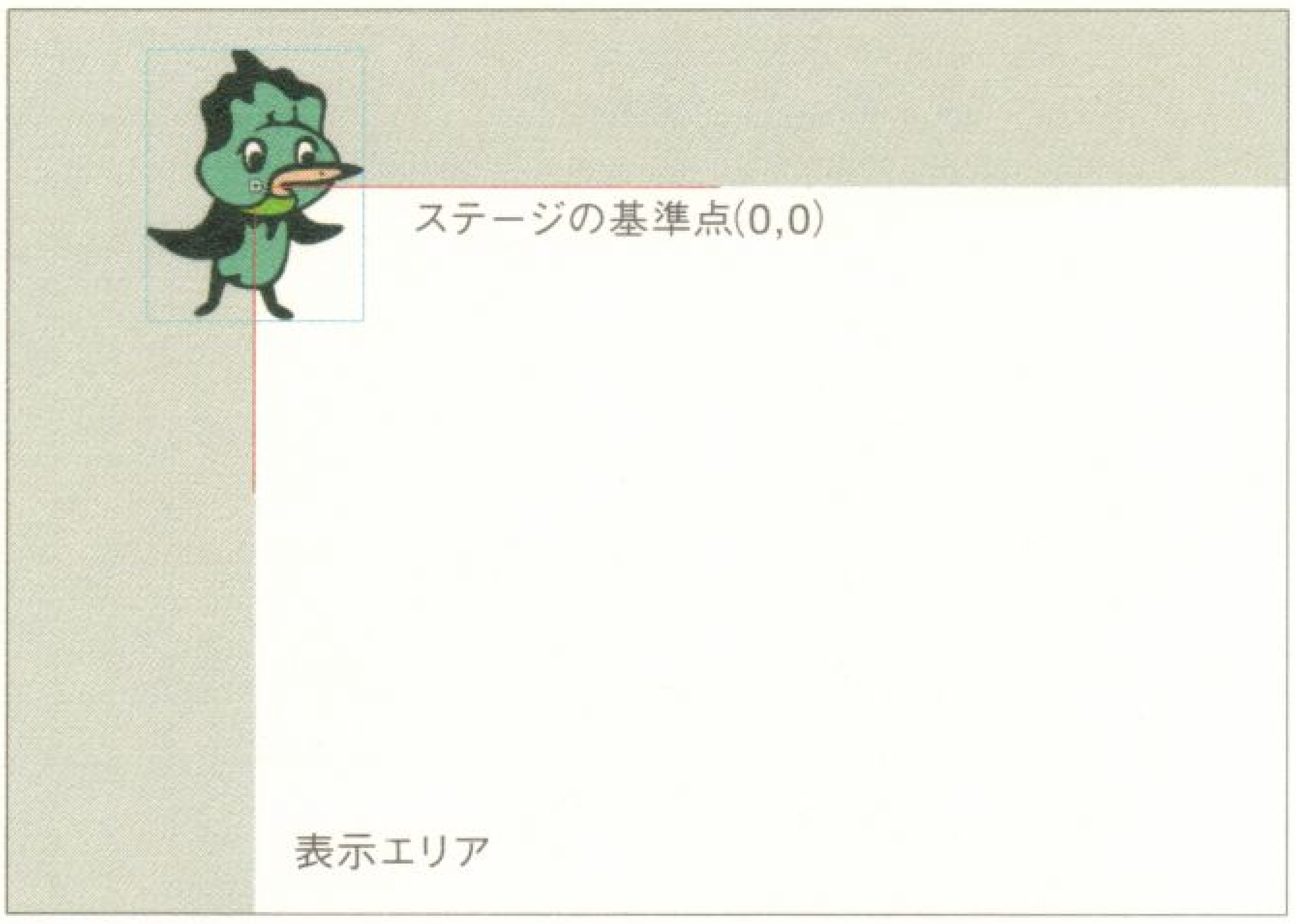
まずは、貼り付けるオブジェクトをインスタンス化（ActionScriptで扱えるように）します。
その後、addChild（貼り付けたいオブジェクト）を指定すると画面上に表示されます。

```
var mc:Kotori = new Kotori();
addChild(mc);
```

しかし、表示されると画面の左隅にキャラクターが見切れた状態で配置されてしまいます。
その理由は基準点と、ポジションにあります。右下の図を見てみましょう。



このように、addChildで配置をすると画面上の左上、つまりステージの基準点にムービークリップの基準点（この場合キャラクターの中心）が配置されます。
それにより、画面の表示エリアにはキャラクターの右下部分のみの表示となっていたわけです。

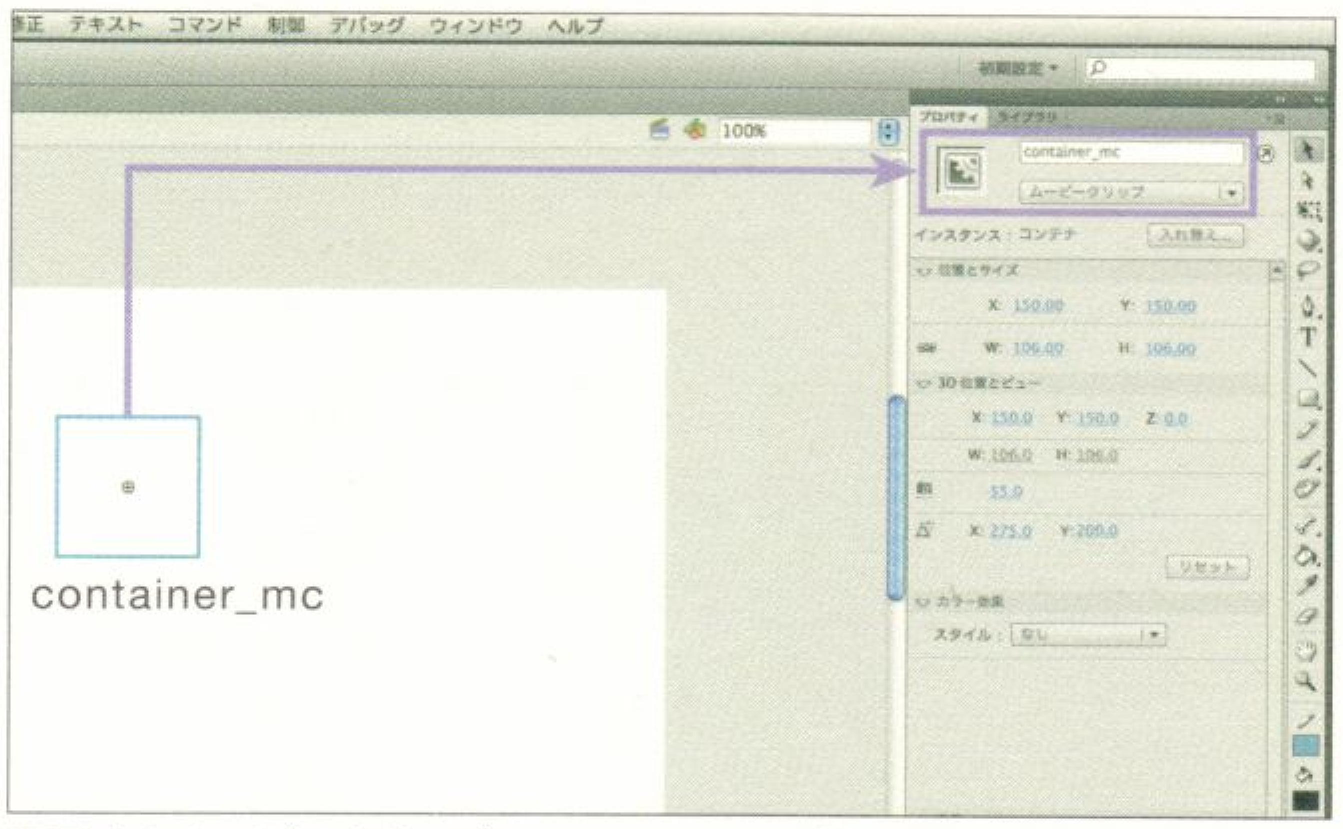


ActionScriptを修正します。このように、配置する際は位置も指定しましょう。

```
//シンボルをインスタンス化する
var mc:Kotori = new Kotori();
//x位置を320(px)にする
mc.x = 320;
//y位置を250(px)にする
mc.y = 250;
//画面上に配置
addChild(mc);
```

ムービークリップの中に配置

addChildを使えば、ステージ上だけではなく、ムービークリップなど別のオブジェクトにも貼り付けが可能です。
右の画面を見てください。
ムービークリップcontainer_mcが画面上にあるとします。



画面上にムービークリップcontainer_mcがある

このcontainer_mcにオブジェクトを貼り付けるには右のように書きます。

```
//シンボルをインスタンス化する
var mc:Kotori = new Kotori();
//container_mcに配置
container_mc.addChild(mc);
```

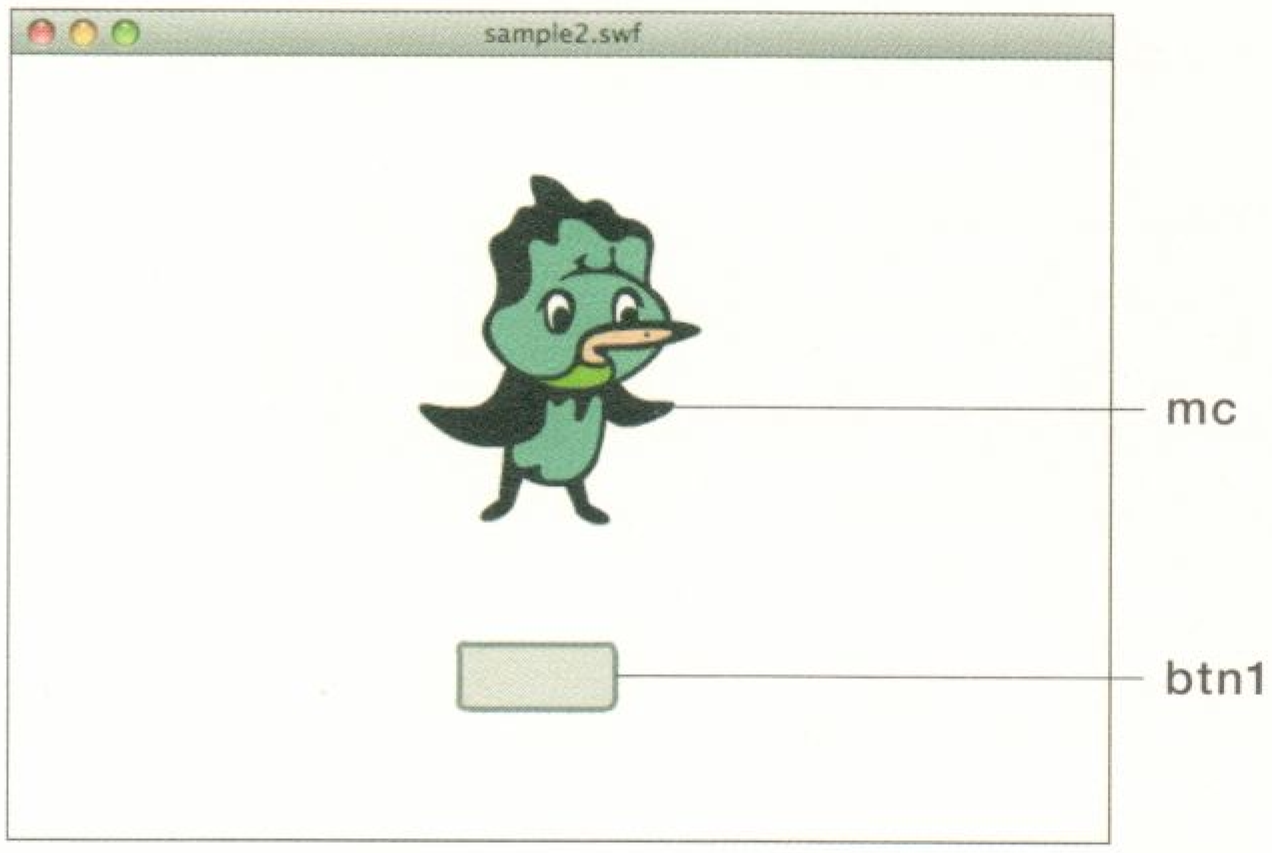
ムービークリップを画面から消す

removeChildを使うことで、addChildで貼り付けたオブジェクトを消すことができます。

使い方

removeChild(消したいオブジェクト)

addChildで画面上に配置したムービークリップ(mc)とボタン(btn1)があります。



ボタンを押すと画面上からキャラクターを消したいので、ActionScriptは右のようになります。

```
//シンボルをインスタンス化する
var mc:Kotori = new Kotori();
//x位置を275(px)にする
mc.x = 275;
//y位置を150(px)にする
mc.y = 150;
//画面上に配置
addChild(mc);


//配置したムービークリップ(mc)を画面上から消すファンクション
function removeMC(ev:MouseEvent):void{
	removeChild(mc);
}

//ボタンが押されるとremoveMCを実行する
btn1.addEventListener(MouseEvent.CLICK, removeMC);
```

LECTURE 4-12

プログラミング［基本］

Object

Objectは複数のデータを保存、受け渡しするためによく使われます。複数のデータを保存するという意味では配列と似ていますが、配列が番号で値を管理するのに対して、オブジェクトは変数名で管理します。

Object

Objectは複数のデータを保持できるが、それぞれのデータに対して名前で管理している。いろいろな種類のデータを管理するのに適している。

住所：新宿
会社：クスール
業種：ウェブ制作
⋮

配列

配列も複数のデータを保持できるが、値を順番で管理している。同じような種類のデータに順番を付けて管理するのに適している。

[0]：東京
[1]：京都
[2]：沖縄
⋮

配列が使われているサンプル

■ PRACTICE 6：
設定XMLファイルを読み込む …… P235へ

合わせて読むと便利な項目

■ MovieClipを制御する ……………… P083へ

Objectの使い方

ObjectはObjectクラスとして提供されています。
他のクラスと同じようにまずはインスタンス化し、それから使用できます。

▶▶ Objectを初期化する

Objectを使うにはObjectクラスを初期化（インスタンス化）する必要があります。

```
var info:Object = new Object();
```

▶▶ Objectに値を入れる

Objectに値を入れるには、「Object.変数＝値」の形式で値を追加します。
1つひとつの値に必ず変数名を付けてください。

```
var info:Object = new Object();
info.address = "新宿";
info.company = "クスール";
info.category = "ウェブ制作";
```

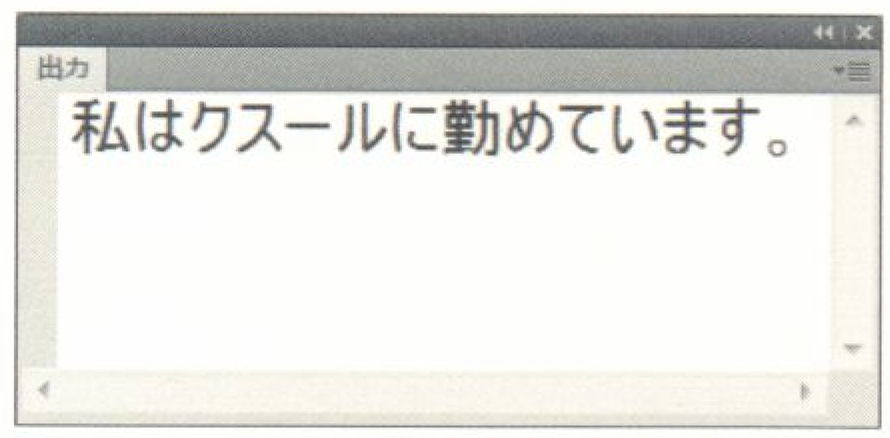
▶▶ Objectから値を取り出す

Objectから値を取り出すには「Object.変数」で取り出します。

```
var info:Object = new Object();
info.address = "新宿";
info.company = "クスール";
info.category = "ウェブ制作";

trace("私は"+info.company+"に勤めています。");
```

実行結果



▶▶ Object内に配列を保存する

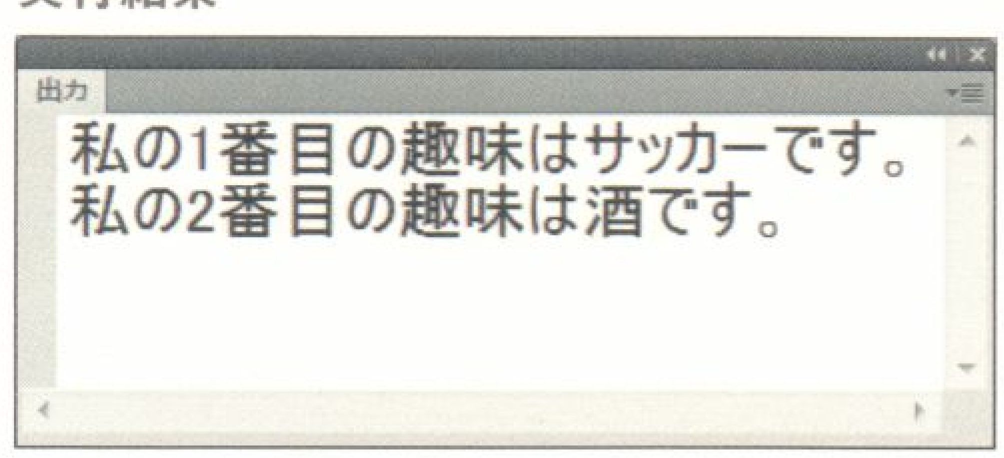
Object内には変数を保存するので、配列の値も変数に保存することができます。
つまりObject内に変数を保存することができます。

■配列 ………………………………………… P115へ

```
var info:Object = new Object();
info.address = "新宿";
..
info.hobby = ["サッカー", "酒"];

trace("私の１番目の趣味は"+info.hobby[0]+"です。");
trace("私の２番目の趣味は"+info.hobby[1]+"です。");
```

実行結果



LECTURE 5-1

プログラミング［応用］

Math関数

ActionScriptでプログラムをする際に知っておきたい項目として、演算子とこのMath関数があります。このMath関数は（Math=数学）というように、主に計算をする上で、非常に便利な機能の1つです。ここでは、よく使うものを中心に紹介します。

Mathが使われているサンプル

■ PRACTICE 3：円の色を変更する… P209へ

合わせて読むと便利な項目

■ trace ………………………………… P071へ
■ 演算子 ………………………………… P101へ
■ 変数 ………………………………… P094へ

主なMath関数

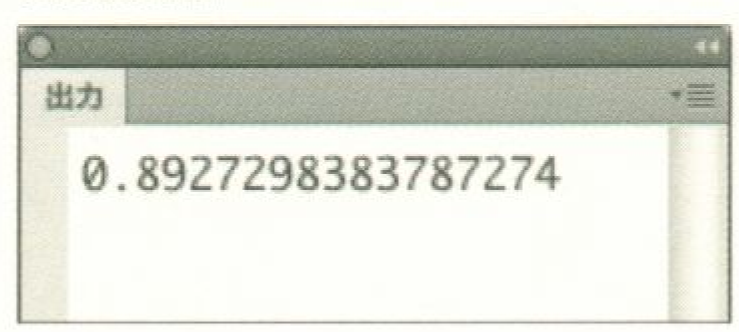
ここで取り上げるのは、ランダム、小数点以下の切り捨てと切り上げ、絶対値を計算するMath関数です。

▶▶ ランダム

Math関数のなかで、よく使われるのがこのランダムでしょう。作品の中でルーレットのようにランダムで結果を表示する場合に、このランダムが利用されています。書き方は右の通りです。
このランダム関数は、0以上1 未満の浮動小数で結果を返します。つまり、どういうことかと言うと、0.000000000000000001から0.999999999999999999までの乱数を結果として返します。この数字を利用して計算に応用するのです。

```
Math.random();
```

実行結果



例えば、ムービークリップ（mc）のx位置をステージ幅550pxの画面上のランダムな位置に貼り付ける場合に、その位置を計算するには右のように書きます。

```
mc.x = Math.random() * 550;
```

実行結果



▶▶ 小数点以下切り捨て（切り上げ）

スゴロクゲームを作って、サイコロの目などの計算をしたい場合、浮動小数では都合が悪い場合があります。そこで、小数を整数に変換する関数を3つ紹介します。これらは場合によって使い分けましょう。

1つ目は小数点切り捨てをする関数。floor（床）なので、下→切り捨てと覚えましょう。

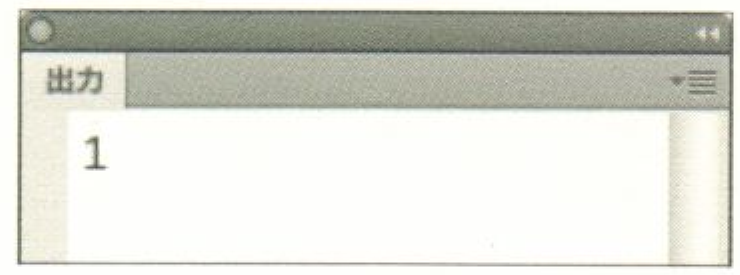
使い方

```
Math.floor(数値または式);
```

使用例

```
trace( Math.floor(1.5) );
```

実行結果



2つ目は小数点切り上げ。ceil（天井）なので、上→切り上げと覚えましょう。

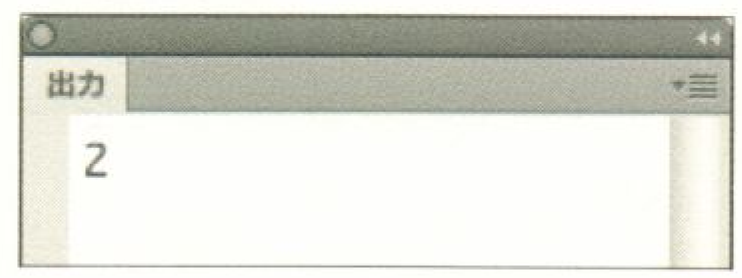
使い方

```
Math.ceil(数値または式);
```

使用例

```
trace( Math.floor(1.5) );
```

実行結果



3つ目は、浮動小数を近い方の整数に変換するラウンド関数です。つまり四捨五入です。

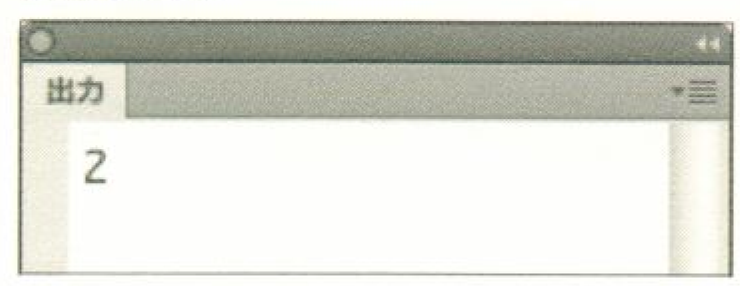
使い方

```
Math.round(数値または式);
```

使用例

```
trace( Math.round(1.5) );
```

実行結果



COLUMN

サイコロの目を計算する式

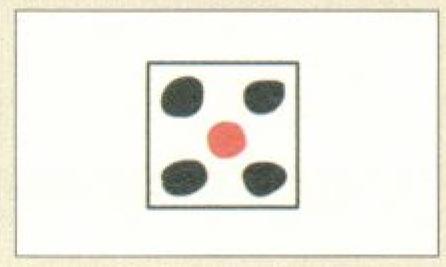
サイコロの目を計算する式を考えてみましょう。サイコロの目が1～6までフレームに並べてある「dice」というインスタンス名を付けたムービークリップを用意します。
ランダムな数字を6とMath.randomで作り、不要な小数をroundで取り払います。このままだと、0～5の数値なので、最後に1を足して1～6の乱数を計算します。

```
trace( Math.round( Math.random() * 6 ) + 1);
```

使用例

```
//1～6までの数字を出す
var num:int = Math.round( Math.random() * 6 ) + 1;
//サイコロの目を変える
dice.gotoAndStop(num);
```

実行結果



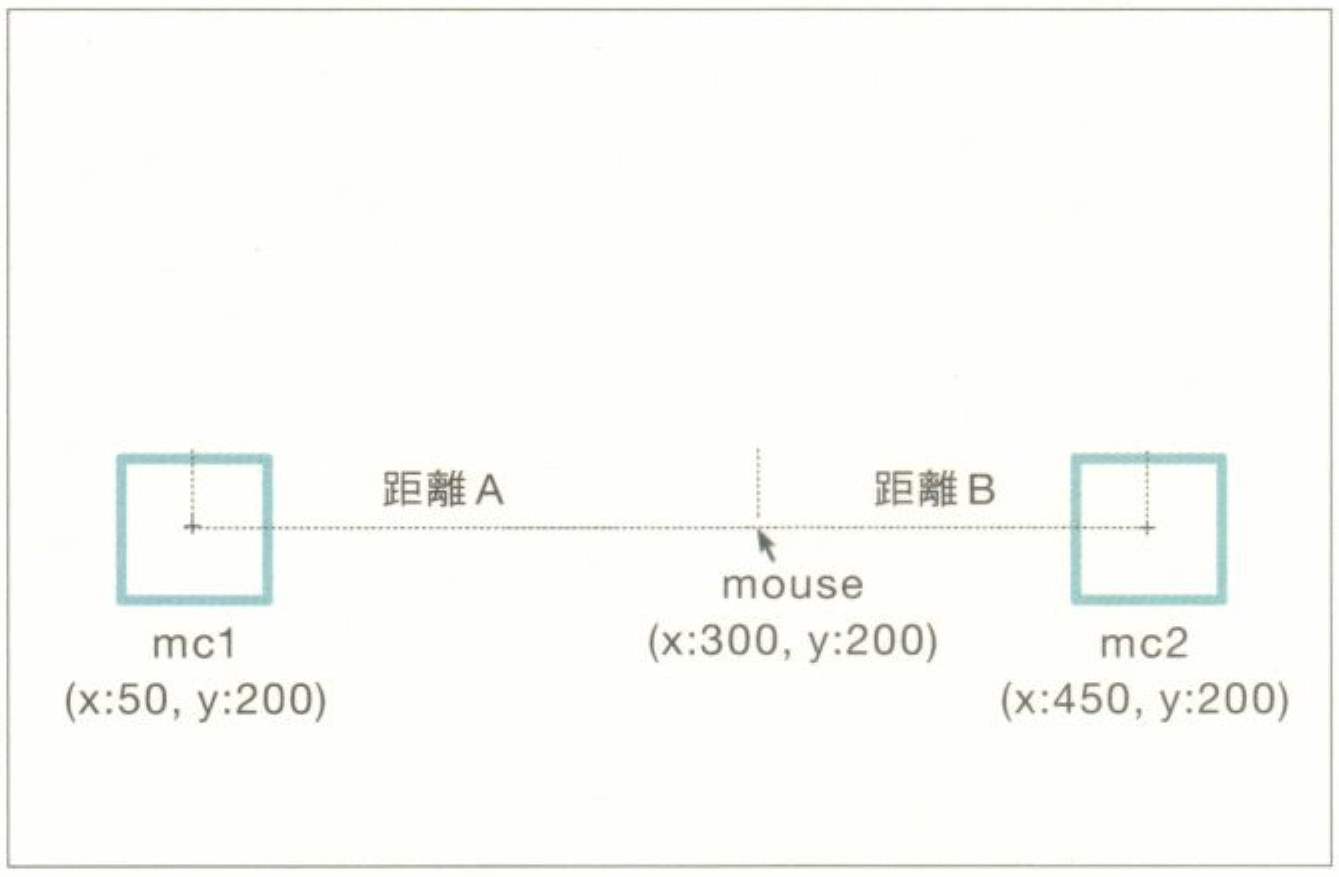
▶▶ 絶対値

絶対値とは、ある数が0からどれだけ離れているかで、数の大きさを計るためのものです。
書き方は右のようになります。

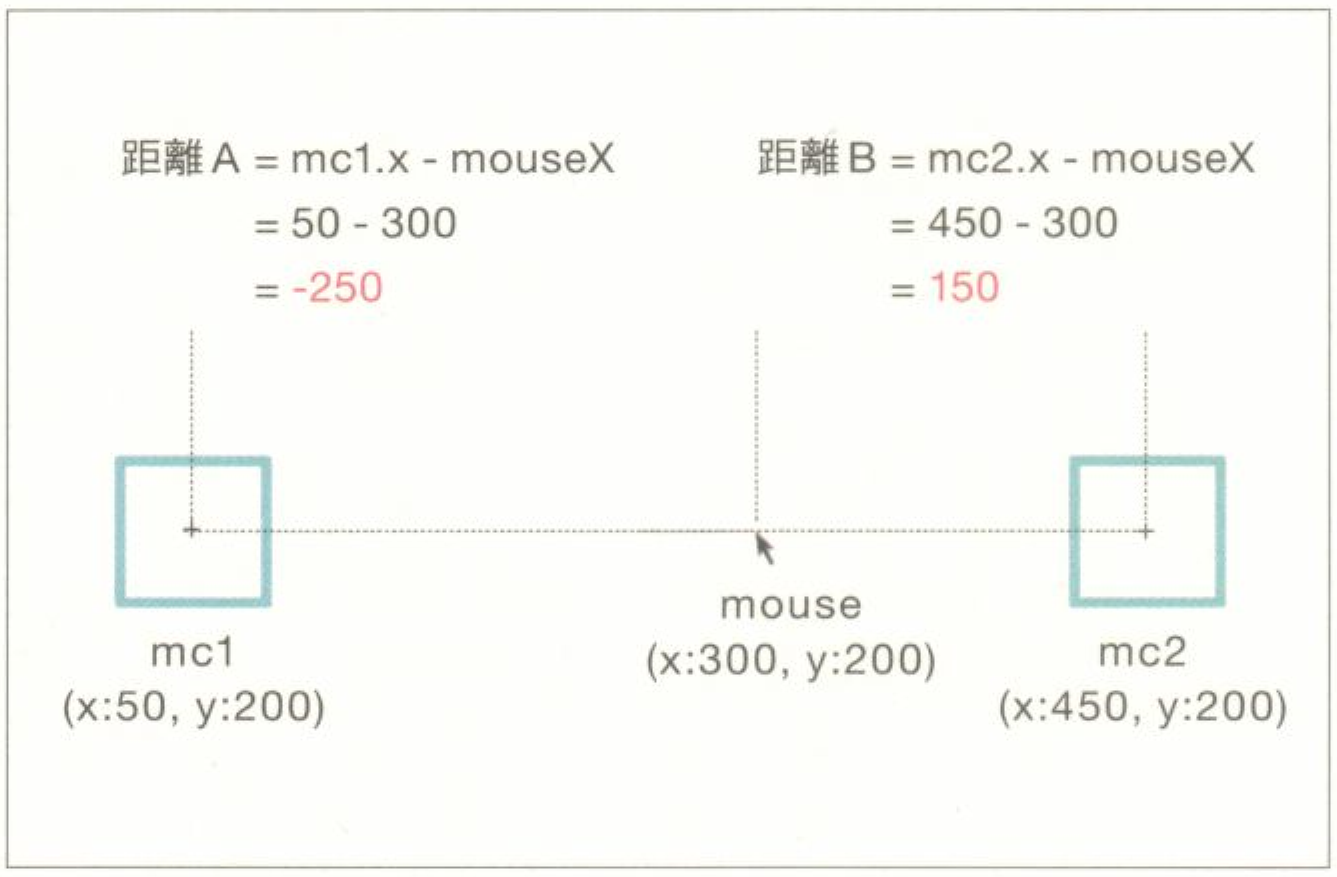
使い方

```
Math.abs(数値または式);
```

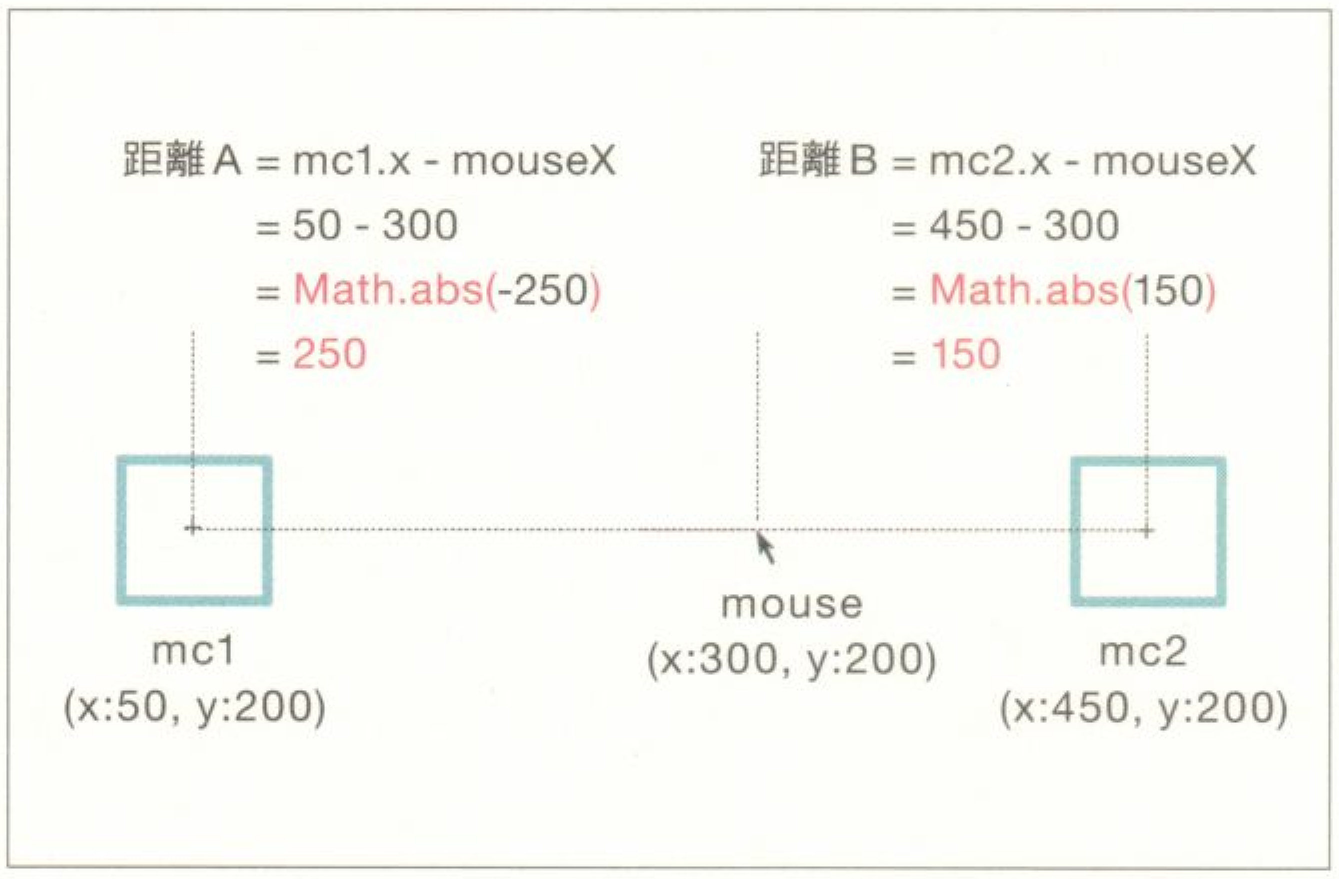
右図のように、マウスから2つのムービークリップ(mc1とmc2)までの距離が、どちらが近いかを判定したい場合、距離の数値の小さい方が近いことになります。



しかし、そのまま計算してしまうと、本来近いはずの距離Bが「150」なのに対し、距離Aが「−250」と、距離Aが小さいと判定されてしまいます。



そこで、その距離を判定するために、この絶対値(Absolute)を使います。



Math関数一覧

その他にも、いくつかあるので使ってみましょう。

パブリック メソッド

Math.random()	ランダムな数を返します。(0.000000000000001 ～ 0.999999999999999)
Math.PI	円周率を返します。(近似値は:3.141592653589793)

小数計算に使うもの

Math.ceil(数値または式)	数値または式を切り上げた値を返します。
Math.floor(数値または式)	数値または式を切り捨てた値を返します。
Math.round(数値または式)	数値または式を最も近い整数に四捨五入し、値を返します。

絶対値を計算

abs(数値または式)	数値または式の絶対値を計算して返します。

三角関数

sin(ラジアン単位の角度)	ラジアン単位で指定された角度のサイン(正弦)を計算して返します。
cos(ラジアン単位の角度)	ラジアン単位で指定された角度のコサイン(余弦)を計算して返します。
tan(ラジアン単位の角度)	指定された角度のタンジェント(正接)を計算して返します。
acos(数値)	数値のアークコサイン(逆余弦)を計算してラジアン単位で返します。
asin(数値)	数値のアークサイン(逆正弦)を計算してラジアン単位で返します。
atan(数値)	数値がタンジェント(正接)の値になる角度を計算してラジアン単位で返します。
atan2(y:Number, x:Number)	y/x 座標の角度をラジアン単位で計算して返します。このとき、円の x 軸(0,0 は円の中心を示します)から反時計回りに測定します。

その他

exp(数値)	自然対数の底(e)を指定された数値(指数)で累乗した値を返します。
log(数値)	数値の自然対数を返します。
max(数値1, 数値2, ... 数値n)	数値1, 数値2(または3つ以上の値)を比べ、一番大きい数を返します。
min(数値1, 数値2, ... 数値n)	数値1, 数値2(または3つ以上の値)を比べ、一番小さい数を返します。
pow(数値1, 数値2)	数値1の数値2乗を計算して返します。
sqrt(数値)	指定された数値の平方根(ルート)を計算して返します。

LECTURE 5-2

プログラミング［応用］

テキストフィールド

Flashで文字を取り扱う場合に利用するのがテキストフィールドです。テキストフィールドは全部で3種類あり、それぞれ用途が違います。

テキストフィールドが使われているサンプル

■ PRACTICE 6：吹き出しの中にタイトルを表示させる ……………………………………P246へ

テキストフィールドの種類と特徴

Flashのテキストフィールドは、
「静止テキスト」「ダイナミックテキスト」「テキスト入力」の3種類があります。

下の表はテキストフィールドの種類と、それぞれの特徴をまとめたものです。
この表からも分かるように、「静止テキスト」はあくまでグラフィック要素として、実行後に文字内容を変えないときに使う設定です。それに対して、「ダイナミックテキスト」と「テキスト入力」は実行後に文字内容を変更する場合に利用します。特に「テキスト入力」はユーザからの文字入力などからの変更も可能とするところが異なる、と理解しておけば良いでしょう。

名称	静止テキスト	ダイナミックテキスト	テキスト入力
フォントを指定して文字表示	できる	できる	できる
文字の選択	できる	できる	できる
実行後の表示文字など変更	できない （グラフィック要素扱い）	できる	できる
フォントの埋め込み	できない	できる	できる
HTML表示	できない	できる	できる
ユーザからの文字入力	できない	できない	できる
ファイルサイズ	軽い	フォント情報を埋め込むと、重くなる可能性あり	フォント情報を埋め込むと、重くなる可能性あり

テキストフィールドの作り方

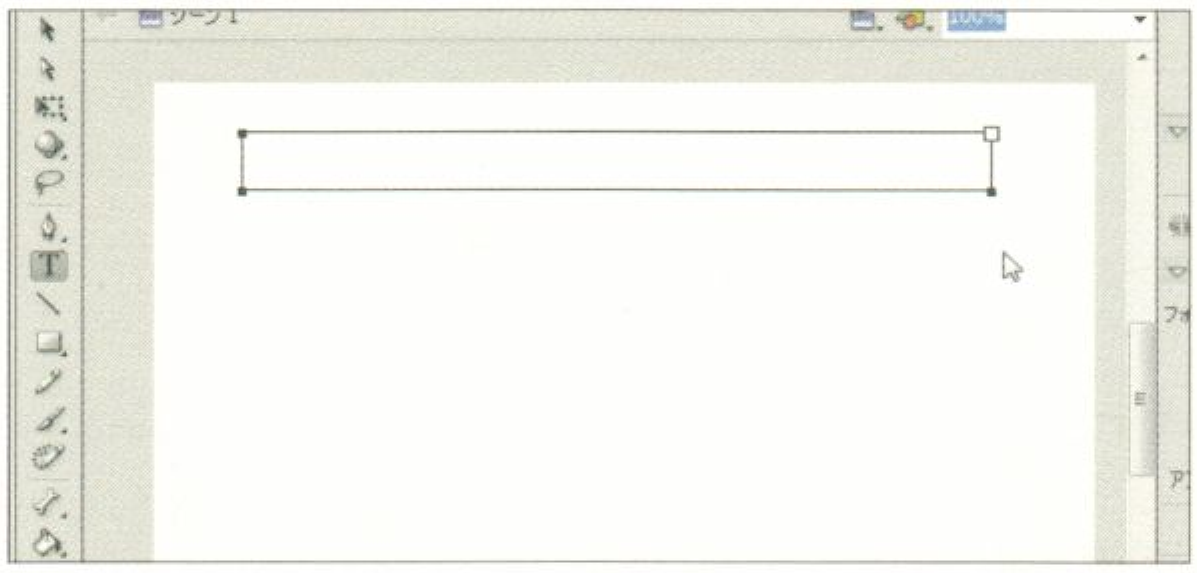
テキストフィールドは、ツールからでも、ActionScriptからでも作成可能です。
ここでは、ツールから作成する場合について紹介します。

1. ツールパネルでテキストツールを選択します。



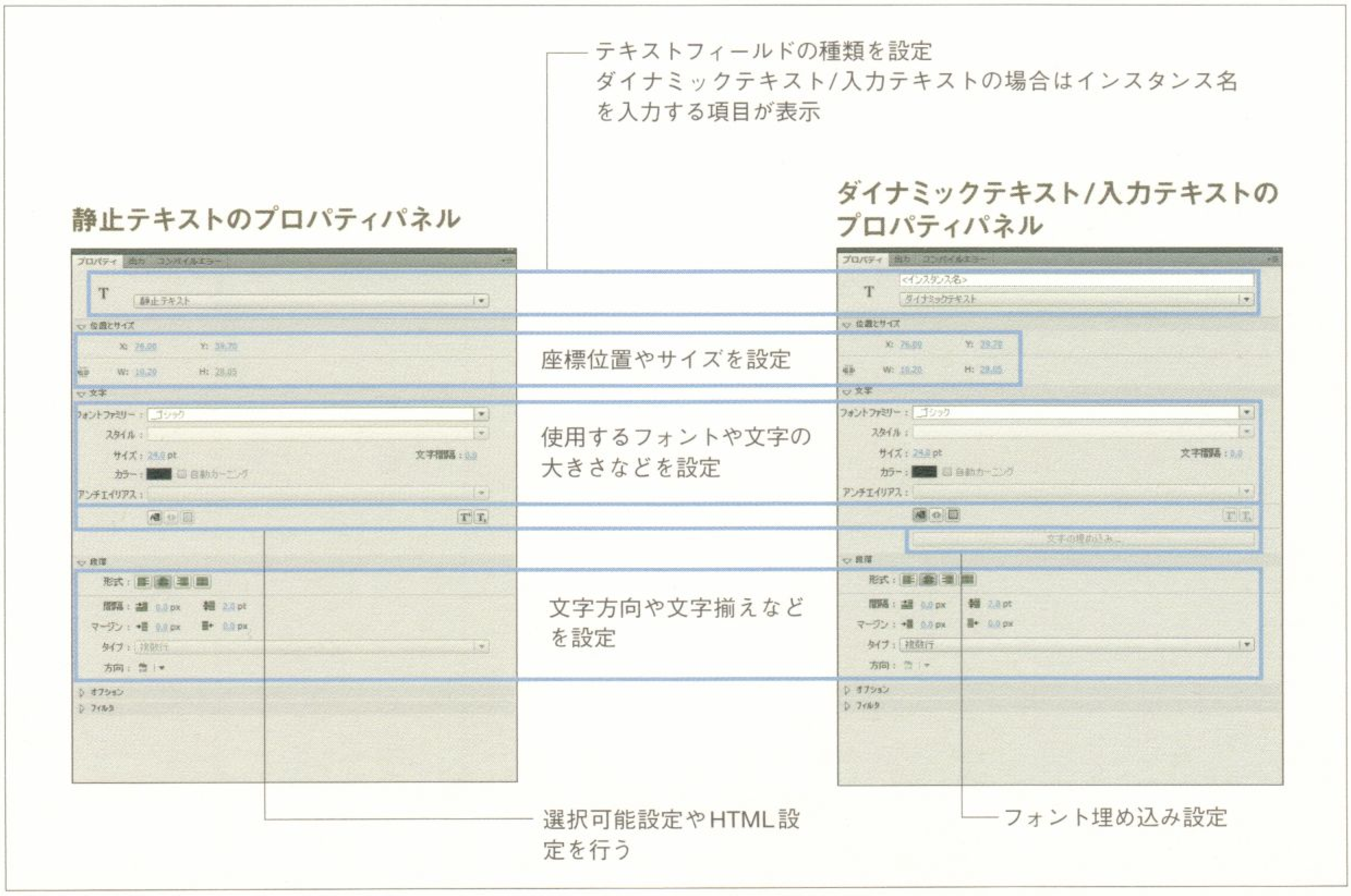
2. ステージ上をクリックする、またはテキストフィールドを設定したい領域をマウスドラッグで指定すると、ステージ上にテキストフィールドが作成されます。





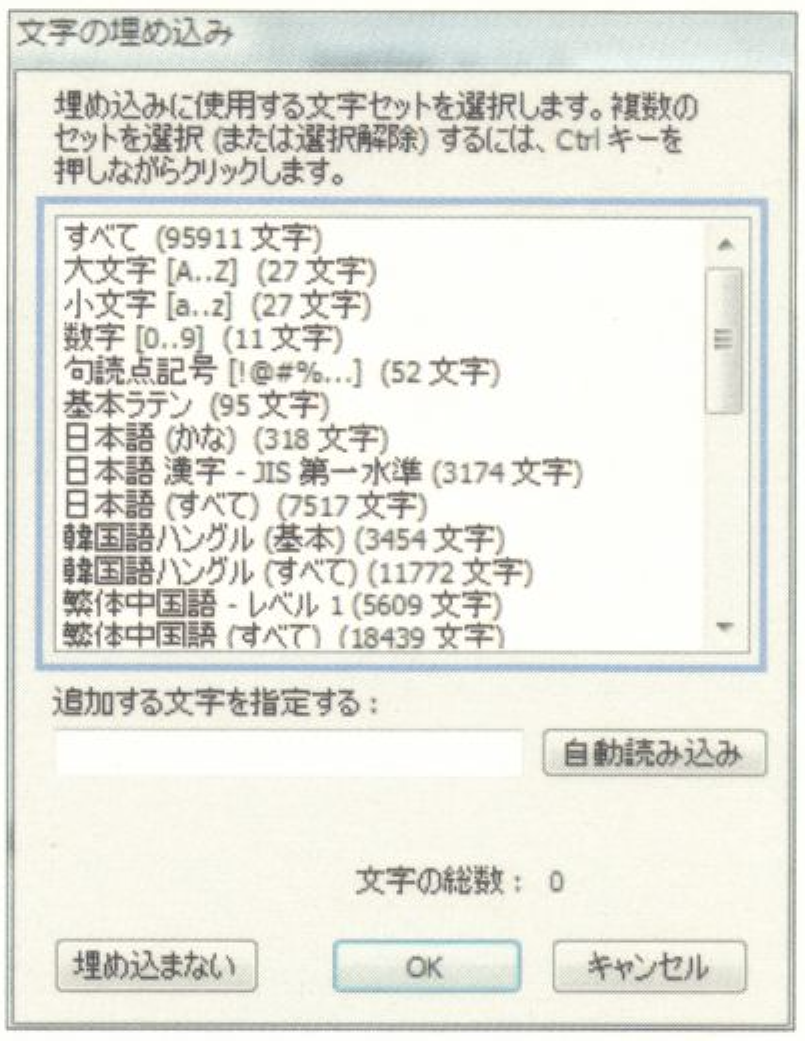
3. 作成したテキストフィールドを選択した状態で、[プロパティ] パネルでテキストフィールドの設定を変更しましょう。前のページで紹介したテキストフィールドの種類も、このパネル上で設定することができます。

3つのタイプごとのパネルの違い



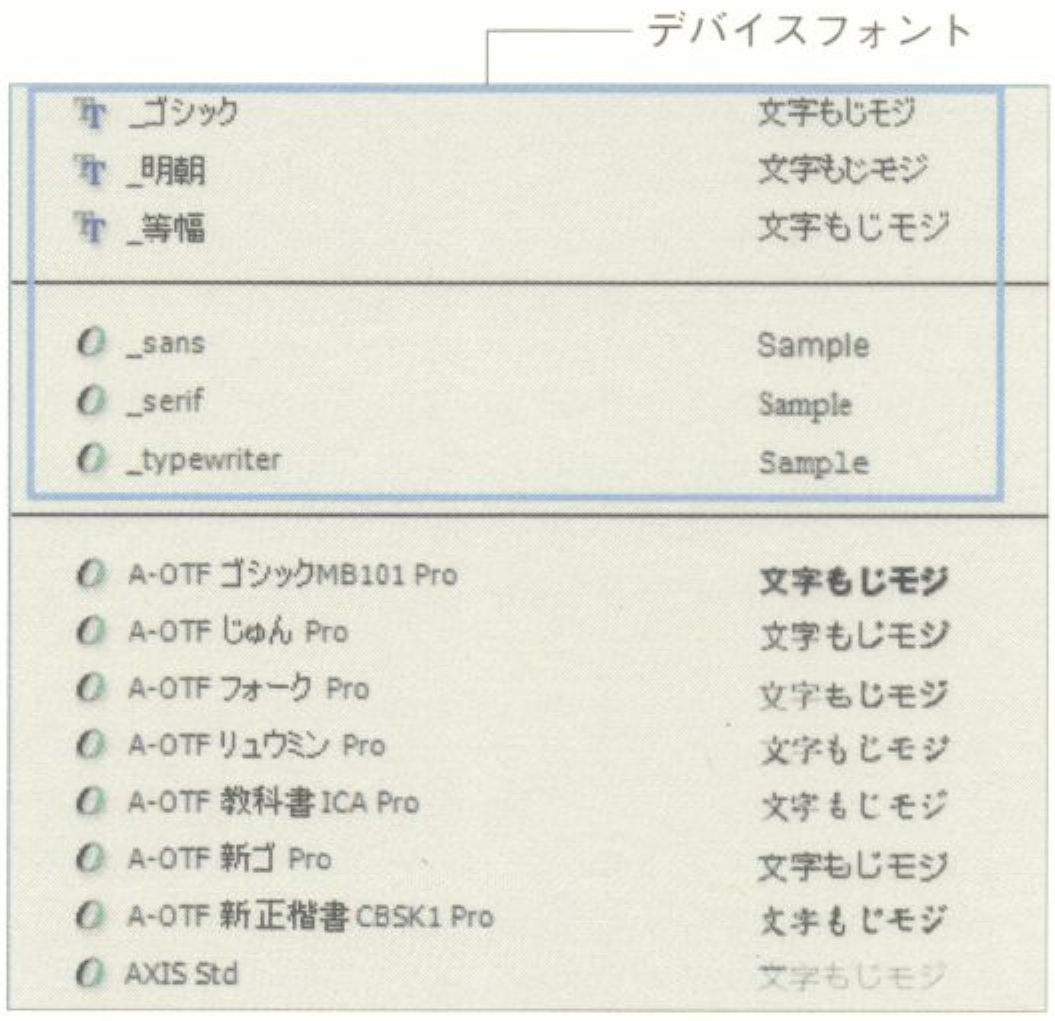
▶▶ フォントの埋め込み

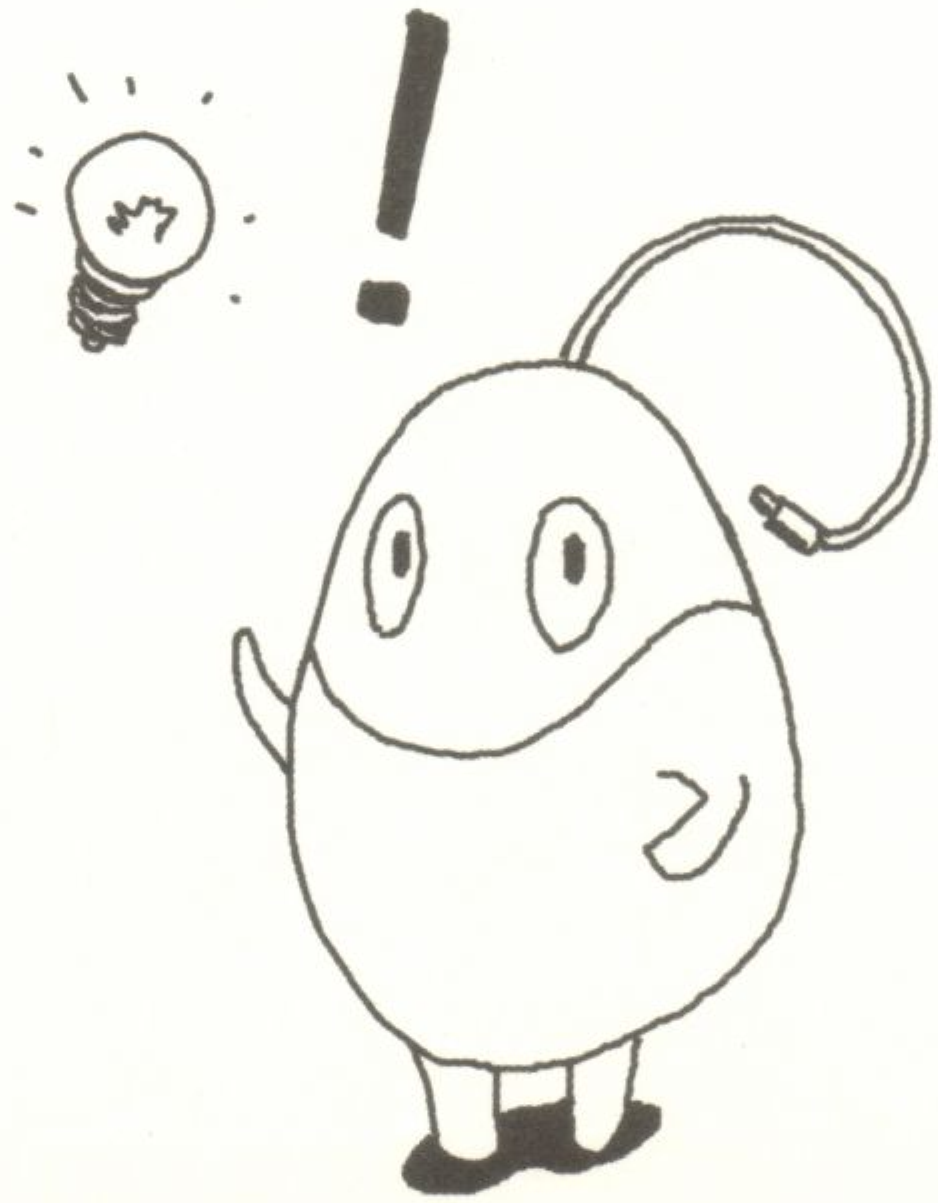
テキストフィールドの種類が「ダイナミックテキスト」もしくは「テキスト入力」の場合、インストールされているフォントデータを埋め込むことができます。フォントを埋め込んでおくと、ここで選択したフォントがインストールされていない環境であっても、意図したフォントで文字が表示されます。
フォントを埋め込む場合は、［プロパティ］パネルの「文字の埋め込み」ボタンを押し、表示される文字の埋め込みダイアログにて埋め込むフォント種類を選択します。
フォント情報は、英語フォントで通常数十KB、日本語フォントでは数MB、パブリッシュ後のSWFファイルサイズが増えるので、必要なフォント(表示される可能性のある文字)のみを埋め込むようにしましょう。



▶▶ デバイスフォント

どんな環境でも正確に表示したい文字を取り扱いたい場合や、フォントを埋め込むことによるSWFファイルサイズの増加を避けたい場合は、［プロパティ］パネルの「フォントファミリー」でデバイスフォントを選びましょう。
デバイスフォントはSWFが実行された環境それぞれに決められているフォントを使うという特殊な設定で、［プロパティ］パネル上ではフォント名の最初に_(アンダーカーソル)が付いているフォントがデバイスフォントに相当します。デバイスフォントは計6種類用意されています。
デバイスフォントを選択すればフォントデータはSWFに埋め込まれないので、出来上がりのSWFファイルサイズが大きくなることはありません。しかし、実行環境によって表示が若干異なる可能性があります。



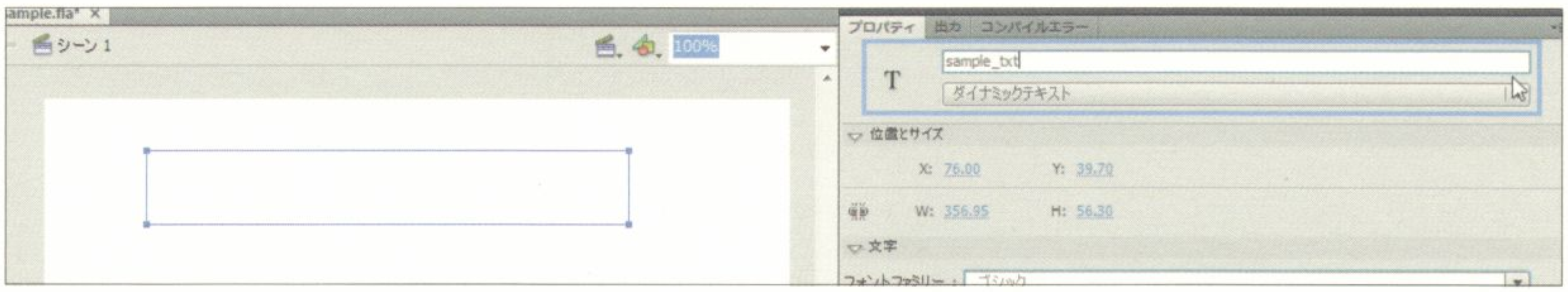


ダイナミックテキスト・テキスト入力の使い方

「ダイナミックテキスト」と「テキスト入力」は、ステージ上に配置するとインスタンス名を設定することができ、プログラムを使ってテキストフィールドをコントロールすることが可能です。また、HTMLを表示させることも可能です。

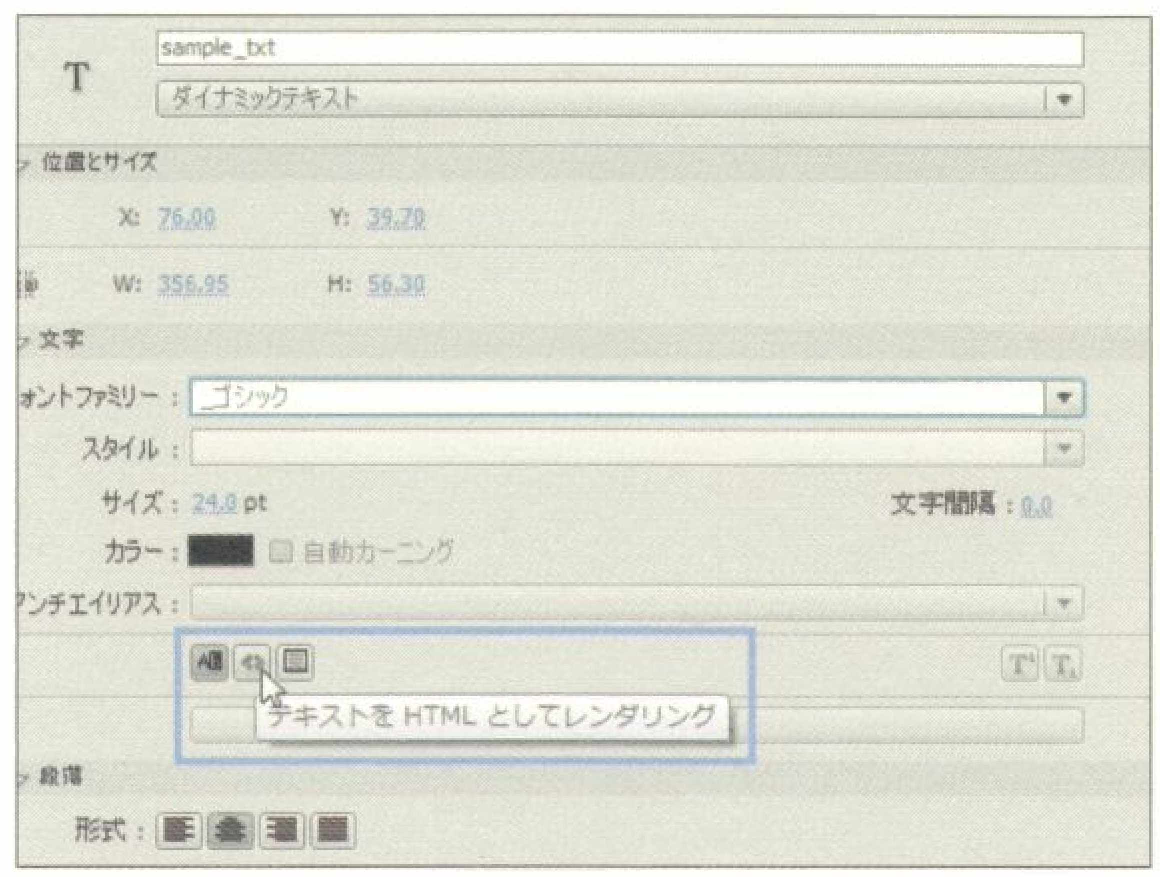
テキストフィールドの設定

テキストフィールド作成後、[プロパティ]パネルにて「ダイナミックテキスト」、もしくは「テキスト入力」を選択すると、[プロパティ]パネル上部にインスタンス名を入力できる項目が表示されるので、インスタンス名「sample_txt」を入力します。



HTMLテキストの設定

ブラウザで表示しているようなHTML形式で情報を表示させたい場合は、「テキストをHTMLとしてレンダリング」ボタンを選択状態にします。
すると、ステージ上に入力した文字をHTMLとして認識し、太字<b>やアンカー<a>などのタグをブラウザのように表示させることができます。
CSSも利用可能です。ただし、ブラウザで使えるすべての機能が有効ではないので、事前に利用可能なタグをチェックしておきましょう。



テキストの表示

ダイナミックテキストやテキスト入力に設定されているテキストフィールドで、現在表示されているテキスト情報を操作したい場合、textプロパティを利用します。
現在表示しているテキスト情報を変更する場合はtextプロパティに新しい文字列を設定し、現在表示しているテキスト情報を取得したい場合もtextプロパティから文字情報を取得します。

```
sample_txt.text = "文字列をセット";
↑ テキストフィールドに文字列を設定する場合
var str = sample_txt.text;
↑ テキストフィールドに表示されている文字列を取得する場合
```

なお、HTML情報はhtmlTextプロパティという、textプロパティとは別のものが用意されているところに注意しましょう。

```
sample_txt.htmlText = "<b>太字文字列をセット</b>";
↑ テキストフィールドにHTMLテキストを設定する場合
var htmlStr = sample_txt.htmlText;
↑ テキストフィールドに表示されているHTMLテキストを取得する場合
```

LECTURE 5-3

プログラミング［応用］

Date

日付や時間などの情報を効率良く取り扱う場合、Dateオブジェクト（Dateクラス）を使うと便利です。Dateオブジェクトは日付や時刻の情報を保持したオブジェクトで、ローカル時間と世界時間（グリニッジ標準時）で日時情報を取り扱うことができます。基本的に日本国内で実行されるようなものを作る場合は、世界時間は意識しなくても問題ないでしょう。特殊な状況でのみ必要な機能です。



合わせて読むと便利な項目

■ クラス ……………………………………………………………P074へ

Dateオブジェクトの使い方

Dateオブジェクトを使うためには、
まずDateクラスをインスタンス化してオブジェクトを作成するところから始めます。

▶▶ Dateオブジェクトの生成

特に引数を与えずにnew演算子にてインスタンスを生成すると、初期化したタイミングの日時情報を持ったDateオブジェクトになります。
なお、日時情報は動作しているOS上での日時情報にセットされますので、もし動作しているOSの日時がずれている場合は、Dateオブジェクトを生成しても日時はずれてしまいます。

```
var date_obj = new Date();
```

特定の日時のDateオブジェクトを生成

引数を設定することもできます。その場合は設定した日時情報を保持したDateオブジェクトが生成されます。
この場合、2009年4月20日13時30分0秒0ミリ秒の日時情報を持ったDateオブジェクトが生成されます。
月は1から12の数字ではなく、0から11の数字で設定する点を注意しましょう。

```
var date_obj = new Date(2009, 3, 20, 13, 30, 0, 0);
⬆ 引数は順に、年、月、日、時、分、秒、ミリ秒、を設定している
```

日時情報の取得

Dateオブジェクトの日時情報を取得します。
Dateオブジェクトの情報を取得するには、Dateオブジェクトのプロパティ(month、hours、secondsなど)から取得するか、DateオブジェクトのgetXXX( )メソッド(getMonth、getHours、getDateなど)から取得するか、2通りの方法があります。この場合、2つのtrace( )の結果は同一になります。

```
trace(date_obj.hours); ⬅ プロパティを使う場合
trace(date_obj.getHours()); ⬅ メソッドを使う場合
```

日時情報の変更

Dateオブジェクトに対して情報を設定する、変更する場合も、Dateオブジェクトのプロパティから設定するか、DateオブジェクトのsetXXX( )メソッド(setMonth、setHours、setDateなど)から設定するか、2通りの方法があります。
この場合も、2つの処理結果は同一になります。

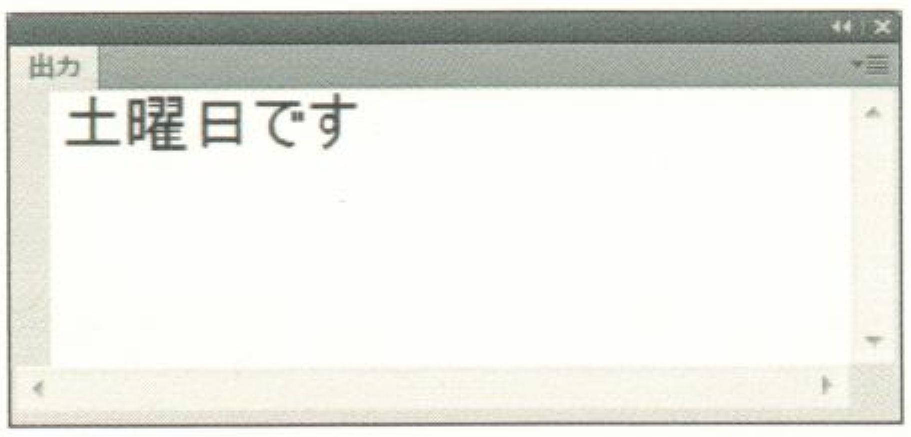
```
date_obj.hours = 6;
date_obj.setHours(6);
```

曜日情報の取得

Dateオブジェクトからは曜日の情報も取得できます。
曜日の情報は0から6までの数字で取得し、それを日本語表記にしたり、英語表記にしたりする作業は、こちらがプログラムを使って行います。
まず日曜日から土曜日までの表記分の要素を持つ配列を用意しておいて、その後に曜日情報を取得し、取得した曜日情報の数値を配列番号として使うことで、シンプルに曜日表記を作成できます。

```
var dayNames = ["日","月","火","水","木","金","土"];
var day = date_obj.day;
trace(dayNames[day] + "曜日です");
```

実行結果



LECTURE 5-4

プログラミング［応用］

Timer

Timer関数は、ActionScript 3.0で追加された新しい機能です。このTimer関数を使うことで、時限爆弾のように、指定したタイミングでプログラムを実行したり、時計のように指定した感覚でプログラムを繰り返し行う処理を作ることができます。

Timerが使われているサンプル

■ PRACTICE 4：Timer関数を使って、画面に配置する ……………………… P216へ

合わせて読むと便利な項目

■ addEventListener ………………… P090へ
■ ENTER_FRAME ……………………… P121へ

Timer関数の使い方

Timer関数には、2種類の使い方が存在します。1つは指定したタイミングでプログラムを実行するとき。もう1つは、一定間隔でプログラムを実行したい場合に利用されます。



Timer関数の使い方

使い方

```
var myTimer:Timer = new Timer(遅延, 繰り返し回数);
```

まずは、Timer関数を使う上で必要な初期設定を見てみましょう。
Timer関数を利用するときは、繰り返す間隔(delay)と繰り返す回数(repeatCount)を設定します。

delay	繰り返す間隔をミリ秒単位(※1000ミリ秒で1秒)で指定します。
repeatCount	繰り返す回数を正数で指定します。何も指定しない場合(デフォルト)は0回で、止めるまで繰り返し続けることができます。

```
//Timer関数の初期設定
var myTimer:Timer = new Timer();
//繰り返す間隔(ミリ秒単位で設定)
myTimer.delay = 1000;
//繰り返す回数
myTimer.repeatCount = 1;
```

また、この初期設定は右のように省略形で書くこともできます。

```
//Timer関数の初期設定 省略形
var myTimer:Timer = new Timer(1000, 1);
```

functionを指定する

Timer関数は、独自で用意した関数を繰り返し実行するのが役割です。
addEventListenerを使用して、呼び出される関数(function)を指定します。
ここでは、独自で用意した「timerHandler」というfunctionを用意しました。
functionの引数には、TimerEventを指定しましょう。また、実行時に何回繰り返しているのかを判別するには、currentCountプロパティを利用しましょう。

```
//繰り返し呼ばれるfunction
function timerHandler(ev:TimerEvent):void{
  //繰り返し呼ばれる処理
  trace("現在の繰り返し回数:" + myTimer.currentCount);
}
//繰り返す関数を指定
myTimer.addEventListener(TimerEvent.TIMER, timerHandler);
```

Timerを開始する

Timer関数は、この開始(start)の合図で始まります。

```
//タイマーを開始
myTimer.start();
```

Timerを停止する

Timer関数を停止(stop)する場合に、使用します。
繰り返し回数を指定していない場合、または任意の理由で停止したい場合に使用します。

```
//タイマーを停止
myTimer.stop();
```

リセットする

一度開始(実行)したTimer関数を、リセット(reset)してはじめからやり直したい場合に使用します。
リセットをした場合は、もう一度開始(start)を使用して、再開します。

```
//タイマーをリセット
myTimer.reset();
```

TIMER_COMPLETEメソッド

Timer関数にはもう1つ便利な機能があります。
繰り返す回数を指定した場合、指定した回数分プログラムが実行され動作が終了すると実行されるのが、この「onComplete」メソッドです。このonComplateメソッドも、動作が終了すると同時に呼び出したいfunctionを指定します。

```
//完了時に呼ばれるfunction
function timerCompleteHandler(ev:TimerEvent):void{
  trace("Timer関数(myTimer)の処理が完了しました!");
}

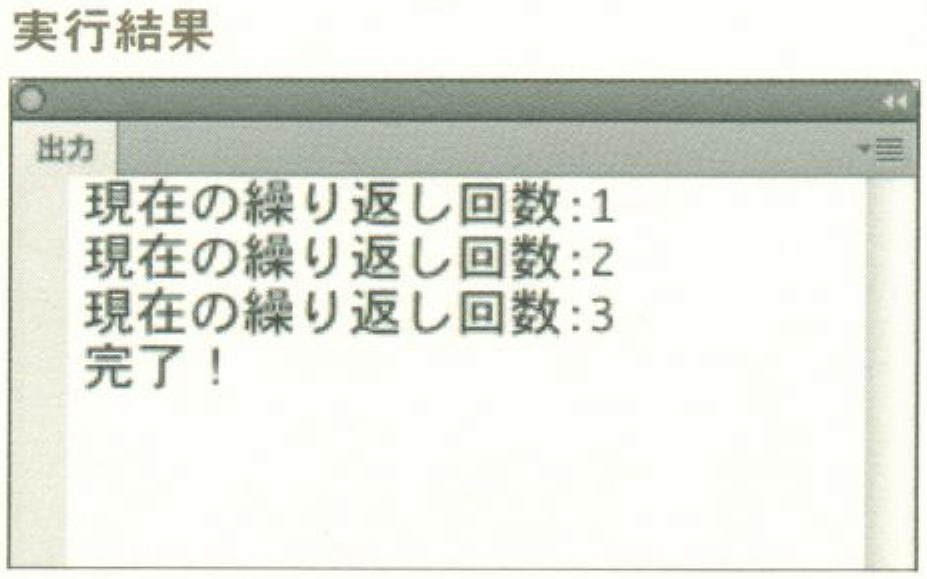
//完了時の処理を指定
myTimer.addEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
```

例えば、3秒カウントダウンして停止するプログラムを作ってみましょう。これらを組み合わせることで、ストップウォッチなど作ってみてはいかがでしょうか。

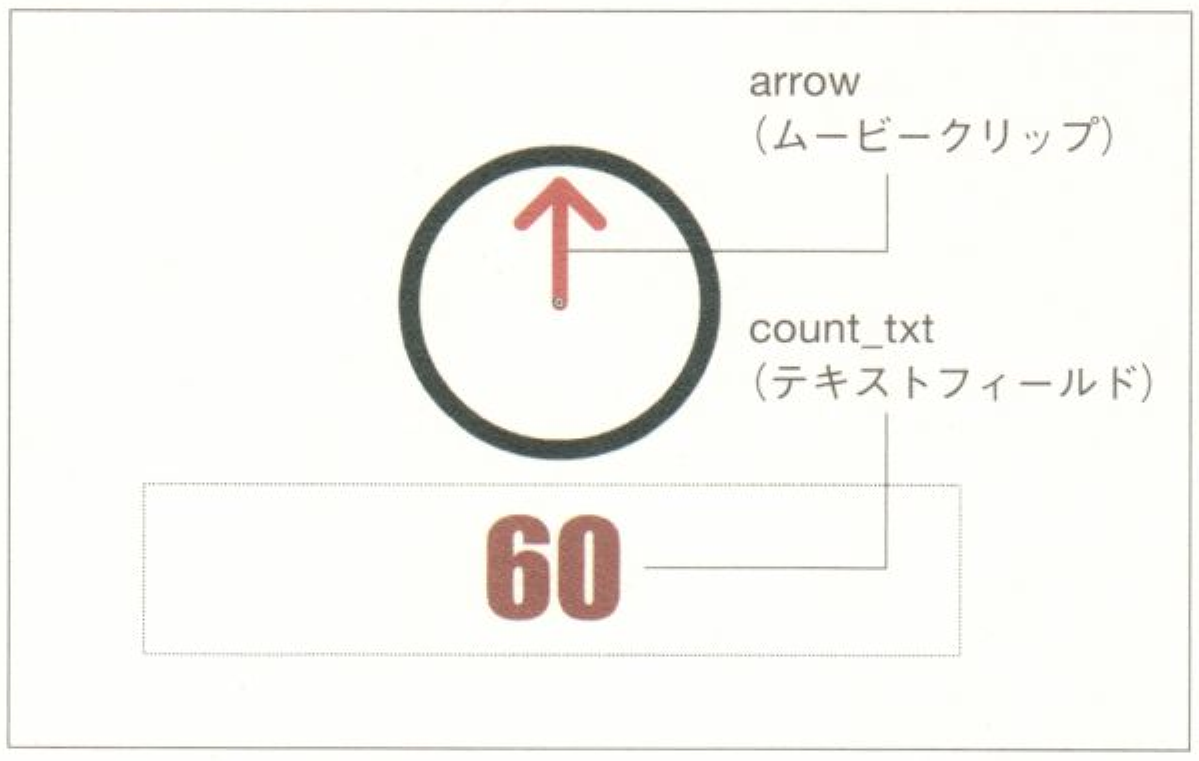
```
//繰り返し呼ばれるfunction
function timerHandler(ev:TimerEvent):void{
  //繰り返し呼ばれる処理
  trace("現在の繰り返し回数:" + myTimer.currentCount);
}
//完了時に呼ばれるfunction
function timerCompleteHandler(ev:TimerEvent):void{
  trace("完了!");
}

//Timer関数の初期設定　1000ミリ秒(1秒)ごとに、3回実行
var myTimer:Timer = new Timer(1000, 3);
//繰り返す関数を指定
myTimer.addEventListener(TimerEvent.TIMER, timerHandler);
//完了時の処理を指定
myTimer.addEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
//開始
myTimer.start();
```

実行結果



▶▶ カウントダウンタイマーの作成



60秒から1秒ずつカウントが減り、最後に「Time UP!!」と表示されるタイマーを作ります。ゲームなどに応用しましょう。

```
//タイマーの数字
var remain:int = 60;

//繰り返し呼ばれるfunction
function timerHandler(ev:TimerEvent):void{
    //1ずつ減らす
    remain--;
    //数字を表示
    count_txt.text = remain.toString();
    //60秒で360度を1周するように回転する
    arrow.rotation -= 360 / 60;
}

function timerCompleteHandler(ev:TimerEvent):void{
    count_txt.text = "TimeUP!!";
}

//Timer関数の初期設定 省略形
var myTimer:Timer = new Timer(100, 60);
//繰り返す関数を指定
myTimer.addEventListener(TimerEvent.TIMER,
   timerHandler);
//完了時の処理を指定
myTimer.addEventListener(TimerEvent.TIMER_
   COMPLETE, timerCompleteHandler);
//開始
myTimer.start();
```

COLUMN

Timer関数での注意点

ものすごく細かい話ですが、このTimer関数でカウントを実行する場合、処理内容によっては、時差が生じることがあります。
ここで紹介したサンプルと同じように、1000ミリ秒ごとに処理を繰り返すfunction(timerHanlder)を指定しました。ここで、もしこの指定した処理がものすごく負荷のかかる処理（例えば、forループ100万回など）であった場合、その処理が終了したタイミングから数えて、1000ミリ秒をカウントするように動くようです。つまり、負担のかかる処理が0.5秒かかった場合、1秒ごとに実行したいのに、1.5秒を要してしまい、結果0.5秒の誤差が生まれてしまうことになります。
それほど重い処理を想定していなければ問題ないと思いますが、正確に動作しているかの検証を行うことをおすすめします。



LECTURE 5-4

LECTURE 5-5

プログラミング［応用］ ステージの情報を取得・設定する

SWF形式で作成されたコンテンツの表示領域全体のことをステージと呼びます。ActionScriptでステージにアクセスすることで、さまざまなステージの情報を取得することができます。取得できる主な情報には以下のようなものがあります。

- ステージの大きさ
- ステージのフレームレート
- 画質
- ステージの配置
- ステージの拡大/縮小モード



Stageクラスが使われているサンプル

■ PRACTICE 7：メニューの位置をステージサイズに合わせ調整する…………… P267へ

合わせて読むと便利な項目

■ イベントの概念 …………………… P077へ

ステージの持っている情報

ここではステージの持っている情報と、
それぞれのActionScriptを使った取得・設定方法を説明します。

▶▶ 取得可能なステージの情報

Stageインスタンスからは主に以下のような情報を取得・設定することができます。

stageWidth

現在のステージの横幅を取得します。

```
trace(stage.stageWidth) // 出力 ( 600 )
```

stageHeight

現在のステージの縦幅を取得します。

```
trace(stage.stageHeight) // 出力 ( 400 )
```

align

ステージの配置位置の指定を取得・設定します。設定値には、StageAlignクラスの持つ定数を使って指定することが可能です。

```
//ステージの配置を左上揃えに設定します
stage.align = StageAlign.TOP_LEFT
```

StageAlignクラスのパブリック定数

定数名	解説
BOTTOM	ステージの配置を下揃えにするよう指定
BOTTOM_LEFT	ステージの配置を左下の隅に揃えるよう指定
BOTTOM_RIGHT	ステージの配置を右下の隅に揃えるよう指定
LEFT	ステージの配置を左揃えにするよう指定
RIGHT	ステージの配置を右揃えにするよう指定
TOP	ステージの配置を上揃えにするよう指定
TOP_LEFT	ステージの配置を左上の隅に揃えるよう指定
TOP_RIGHT	ステージの配置を右上の隅に揃えるよう指定

frameRate

ステージのフレームレートを取得・設定します。

```
trace(stage.frameRate) //出力(24)
```

quality

ステージのレンダリング品質を取得・設定します。設定値には、StageQualityクラスの持つ定数を使って指定することが可能です。

```
//ステージの画質を最高に設定
stage.quality = StageQuality.BEST;
```

StageQualityクラスのパブリック定数

定数名	解説
BEST	非常に高いレンダリング品質を指定
HIGH	高いレンダリング品質を指定
LOW	低いレンダリング品質を指定
MEDIUM	通常のレンダリング品質を指定

scaleMode

ステージの拡大/縮小モードを取得・設定します。設定値には、StageScaleModeクラスの持つ定数を使って指定することが可能です。

```
//ステージの拡大・縮小モードをサイズを維持するように設定
stage.scaleMode = StageScaleMode.NO_SCALE;
```

StageScaleModeクラスのパブリック定数

定数名	解説
EXACT_FIT	表示領域内にステージ全体が、元の縦横比を維持しないで表示されるよう指定
NO_BORDER	表示領域いっぱいにステージ全体が歪まずに表示されるよう指定
NO_SCALE	ステージのサイズが固定され、ウィンドウのサイズが変更された場合でも、サイズが維持されるように指定
SHOW_ALL	表示領域内に、ステージ全体が元の縦横比を維持したまま、歪まずに表示されるよう指定

ステージサイズの変化に応じて情報を取得する

ActionScriptを使ってステージサイズの変化のタイミングを知ることができます。
ここでは、ステージサイズの変化に応じてリアルタイムにステージのサイズを表示する方法を説明します。

▶▶ ステージのサイズの変化を受け取る

ステージのサイズが変わった際、StageインスタンスがEvent.RESIZEを発行します。
このイベントをリスナーに登録することで、ステージのサイズの変化を監視することが可能になります。

```
stage.addEventListener(Event.RESIZE,onResizeStage);

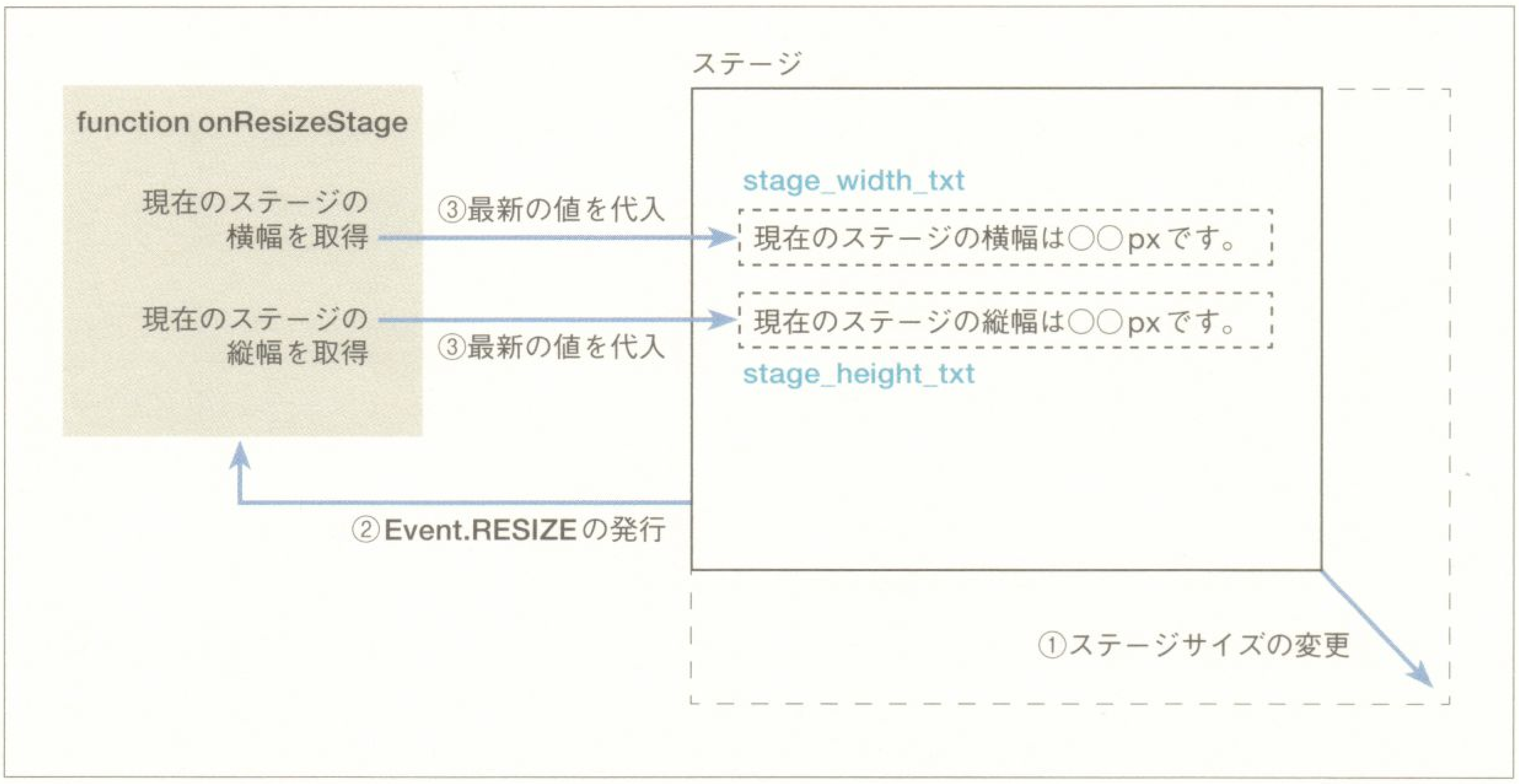
function onResizeStage(event){
  trace("ステージのサイズが変わりました");
}
```

出力結果

```
出力
ステージのサイズが変わりました
```

▶▶ サンプル：ステージの大きさを表示する

現在のステージの大きさを、画面上に表示するサンプルを作成します。



1. ステージの大きさが変更されると、Event.RESIZEが発行されます。
2. Event.RESIZEを受け取り、ステージの最新のサイズを取得します。
3. ステージの最新のサイズをテキストフィールドに表示します。

```
stage.align = StageAlign.TOP_LEFT;
stage.quality = StageQuality.BEST;
stage.scaleMode = StageScaleMode.NO_SCALE;

stage.addEventListener(Event.RESIZE,onResizeStage);

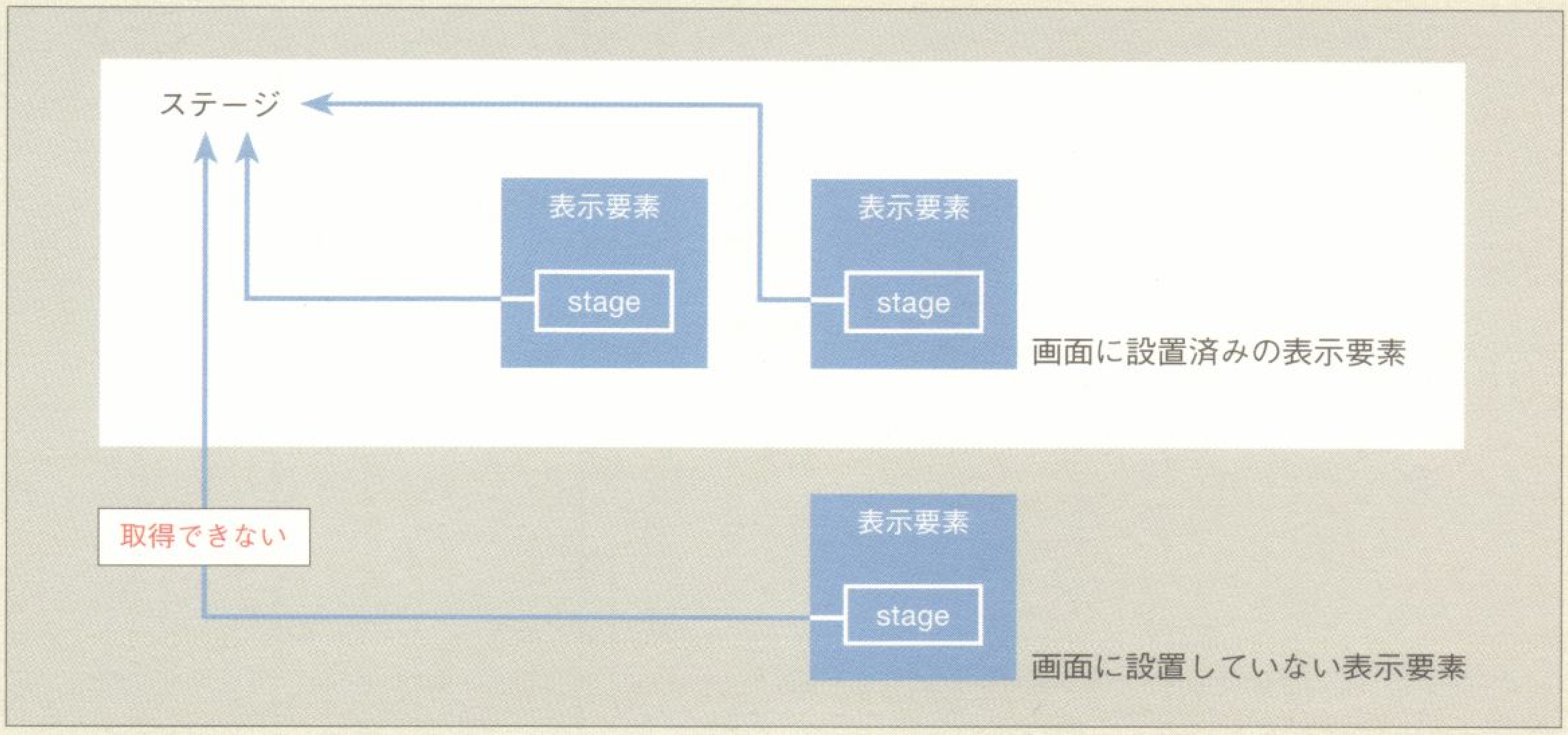
function onResizeStage(event){
    var now_stage_width = stage.stageWidth; ⬅ 最新のステージの横幅を取得
    stage_width_txt.text = "現在のステージの横幅は" + now_stage_width + "pxです。";

    var now_stage_height = stage.stageHeight; ⬅ 最新のステージの縦幅を取得
    stage_height_txt.text = "現在のステージの縦幅は" + now_stage_height + "pxです。";
}
```


COLUMN

ステージへのアクセス

ステージの情報の取得・設定はStageインスタンスを用います。
SWF上のStageインスタンスへのアクセスは、各タイムラインからstageプロパティを参照することで可能です。stageプロパティは表示要素のクラスそれぞれが持っていて、画面に配置された状態で参照することができます。



LECTURE 5-6

プログラミング［応用］

mouseX・mouseY

Flashでは、画面上のマウスの位置を取得することができます。マウスの位置により画面に変化を付けたり、マウスに追従するキャラクターを作ったり、独自のマウスポインタを作ったりと、さまざまな用途でこのマウスの位置を利用します。

マウスの位置が使われているサンプル

■ PRACTICE 3：
マウスの位置に円を表示する……… P208へ

合わせて読むと便利な項目

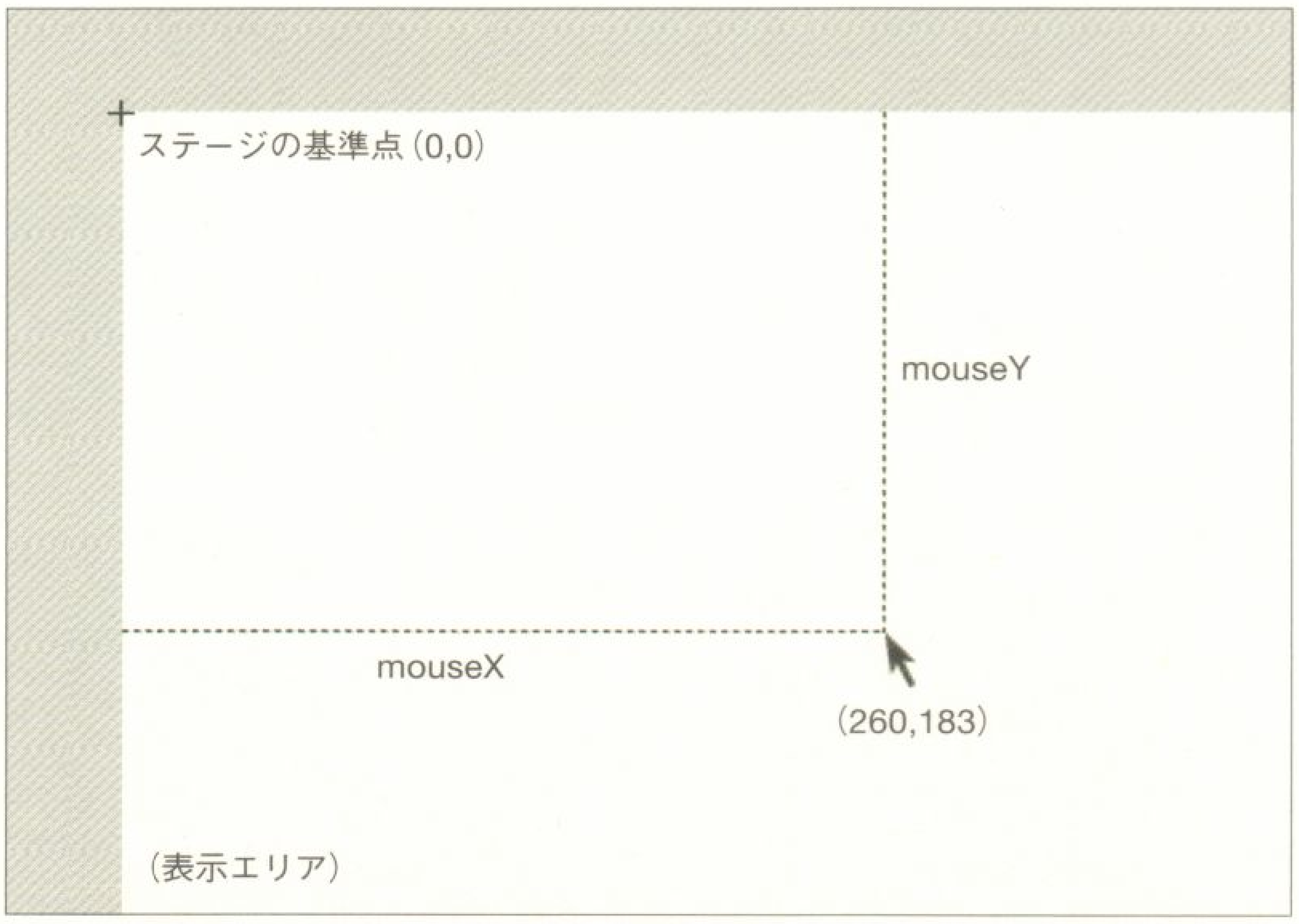
■ ENTER_FRAME …………………… P121へ

マウスの位置の取得

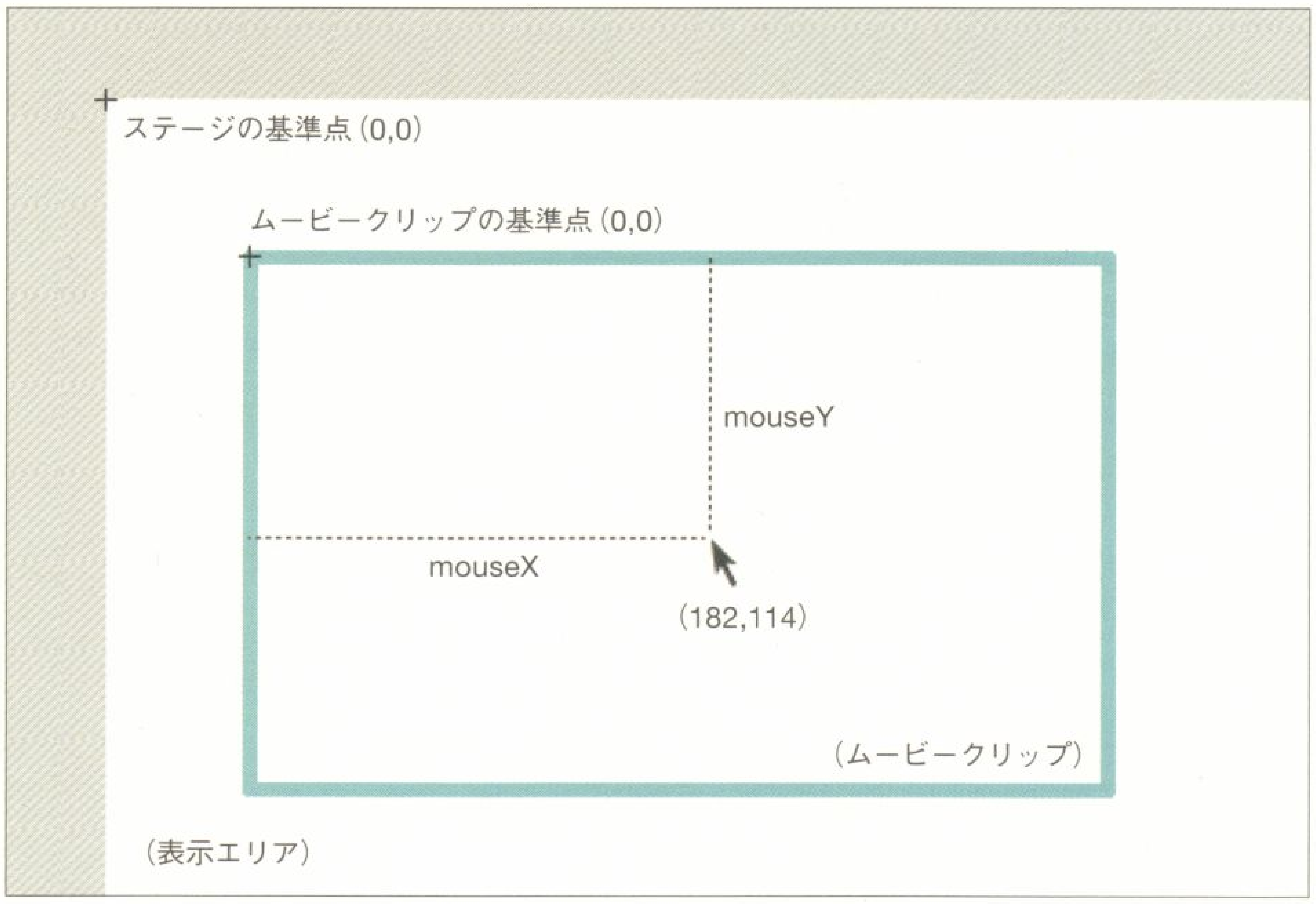
マウスの位置はX軸とY軸の2点で特定しますが、
メインタイムラインかムービークリップの中かによっても計り方が変わってきます。

▶▶ マウスの位置の考え方

マウスの位置は、X軸（横軸）とY軸（縦軸）の2点で特定します。
位置の計り方は、図のようにメインタイムラインの表示エリアの左上を原点とし、ピクセルで取得することができます。
この図では、マウスのX位置は260になり、マウスのY位置は183になります。



ムービークリップの場合も同様に基準点から計測した2つの点で特定することができます。
この図では、ムービークリップの基準点を元に取得するので、マウスのX位置は182になり、マウスのY位置は114になります。



▶▶ マウスの位置の取得方法

マウスの横軸を取得

```
mouseX
```

マウスの縦軸を取得

```
mouseY
```

マウスの位置を取得する方法を、いくつかの場合に分けて紹介します。

■メインタイムライン上からマウスの位置を取得

□メインタイムラインのマウスの位置を取得

```
//マウスの位置を出力する
trace(" マウスのX軸:" + mouseX);
trace(" マウスのY軸:" + mouseY);
```

□ムービークリップ(インスタンス名:mc)内のマウスの位置を取得

これを[mouseX2.fla]のように、メインタイムラインのactionレイヤーを見てください。
ENTER_FRAMEとtraceを使って、常にマウスの位置を計測できるプログラムを用意しました。

```
//ムービークリップ内のマウスの位置を出力する
trace(" マウスのX軸:" + mc.mouseX);
trace(" マウスのY軸:" + mc.mouseY);
```

■ムービークリップの中からマウスの位置を取得

□ムービークリップ内のマウスの位置を取得

「□メインタイムラインのマウスの位置を取得」で使用したものと同じActionScriptをムービークリップ内のタイムラインに記述すると値が取得できます。

```
//マウスの位置を出力する
trace(" マウスのX軸:" + mouseX);
trace(" マウスのY軸:" + mouseY);
```

□ムービークリップの中から、1階層上のマウスの位置を取得

これを［mouseX3.fla］のように、メインタイムライン上のムービークリップmcの中のactionレイヤーを見てください。
ENTER_FRAMEとtraceを使って、常にマウスの位置を計測できるプログラムを用意しました。

```
//ムービークリップの1階層上のマウスの位置を
  出力する
trace(" マウスのX軸:" + parent.mouseX);
trace(" マウスのY軸:" + parent.mouseY);
```

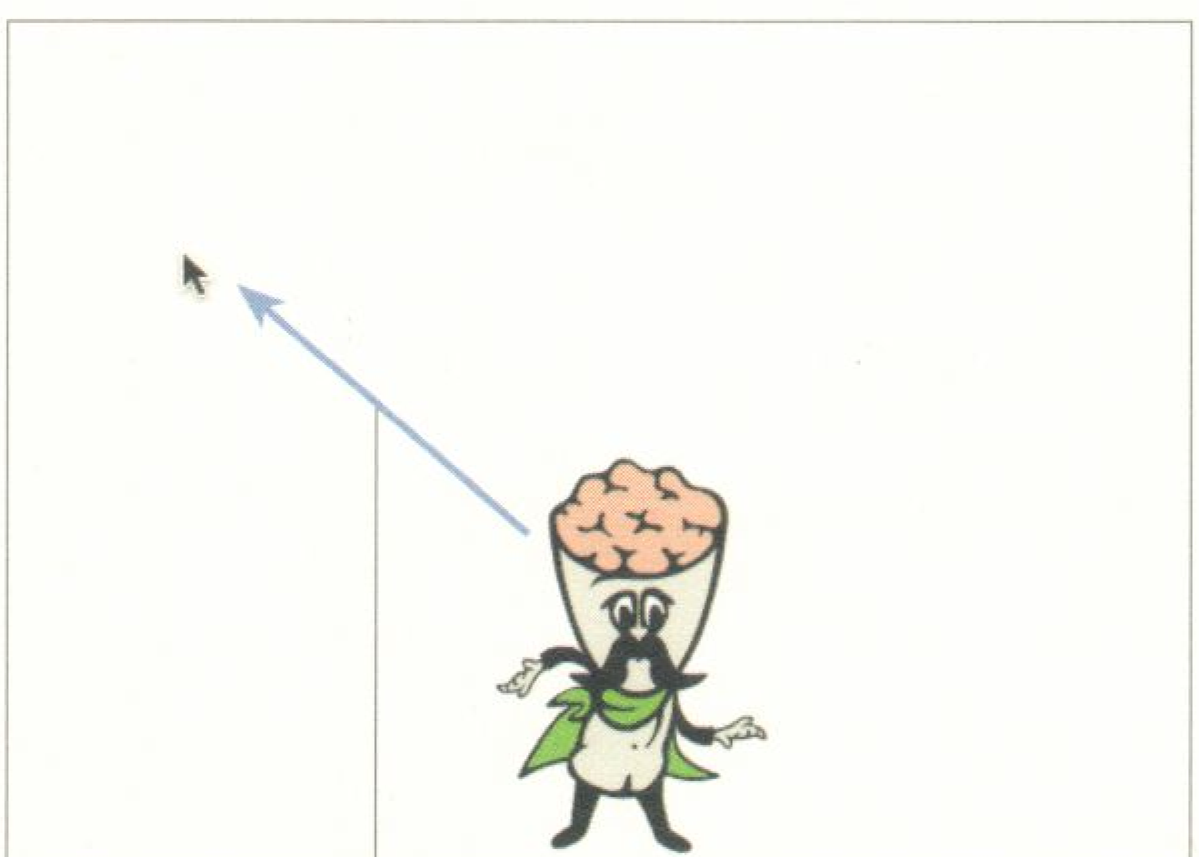
□ムービークリップの中から、メインタイムラインのマウスの位置を取得

また、ムービークリップが複数の入れ子状態である場合、たとえば、2階層以上入れ子の中のムービークリップからメインタイムライン上のマウスの位置を取得する場合は、このように「root」というプロパティを使うと、メインタイムライン上のマウスの位置が取得できます。

```
//メインタイムライン上のマウスの位置を出力する
trace(" マウスのX軸:" + root.mouseX);
trace(" マウスのY軸:" + root.mouseY);
```

▶▶ サンプル：マウスについてくるキャラクター

mouseXとmouseYを使って、マウスについてくるキャラクターのサンプルを作りましょう。



マウスを移動させると、キャラクターもついて移動する

サンプル例

```
//繰り返し呼ばれるfunction
function mousePos(ev:Event){
   //マウスの位置にキャラクターを表示
   chara.x = mouseX;
   chara.y = mouseY;
}

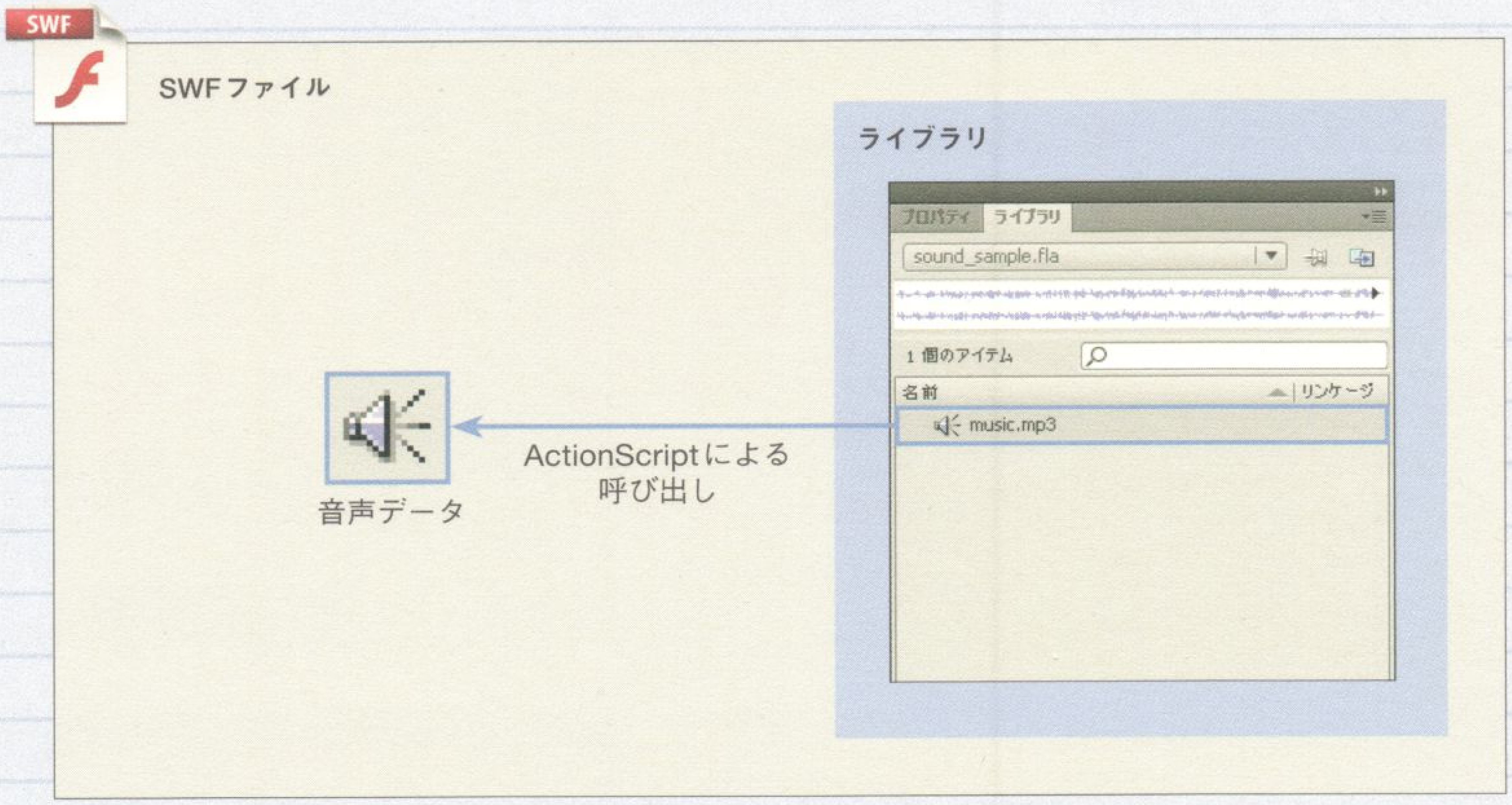
//繰り返す関数を指定
addEventListener(Event.ENTER_FRAME, mousePos);
```

LECTURE 5-7

プログラミング［応用］

ActionScriptでのサウンドの制御

Flashではサウンドをタイムラインに埋め込むだけでなく、サウンドを動的に制御することもできます。ライブラリ内の音声データをリンケージ設定しておくことで、任意のサウンドをActionScriptで呼び出すことが可能になります。



Soundクラスが使われているサンプル

- PRACTICE 8：
  サウンドを制御する…………………… P279へ

合わせて読むと便利な項目

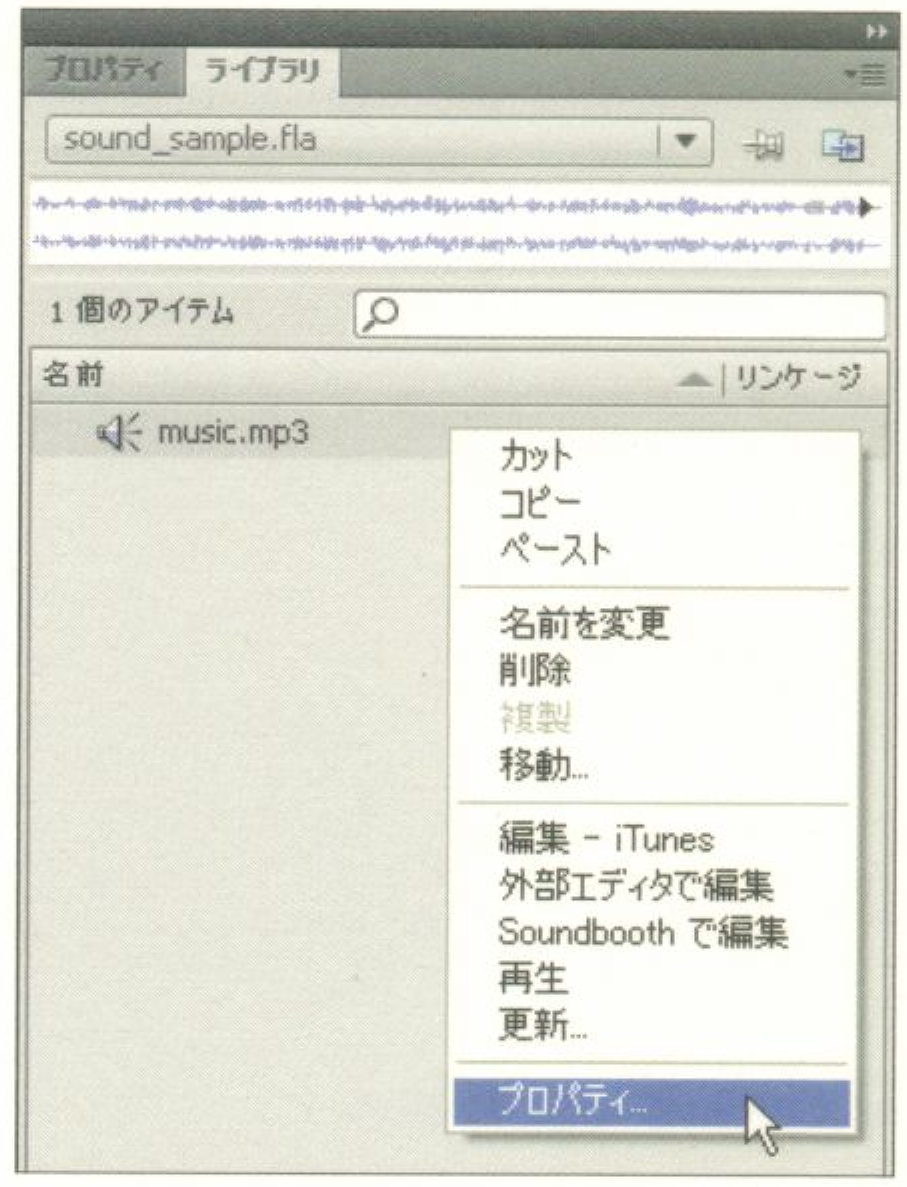
- サウンドの扱い方 …………………… P056へ
- Loader ………………………………… P158へ

音声アイテムのリンケージ設定を行う

ライブラリ内のサウンドデータをActionScriptで呼び出せるようにするために、
リンケージ設定を行います。

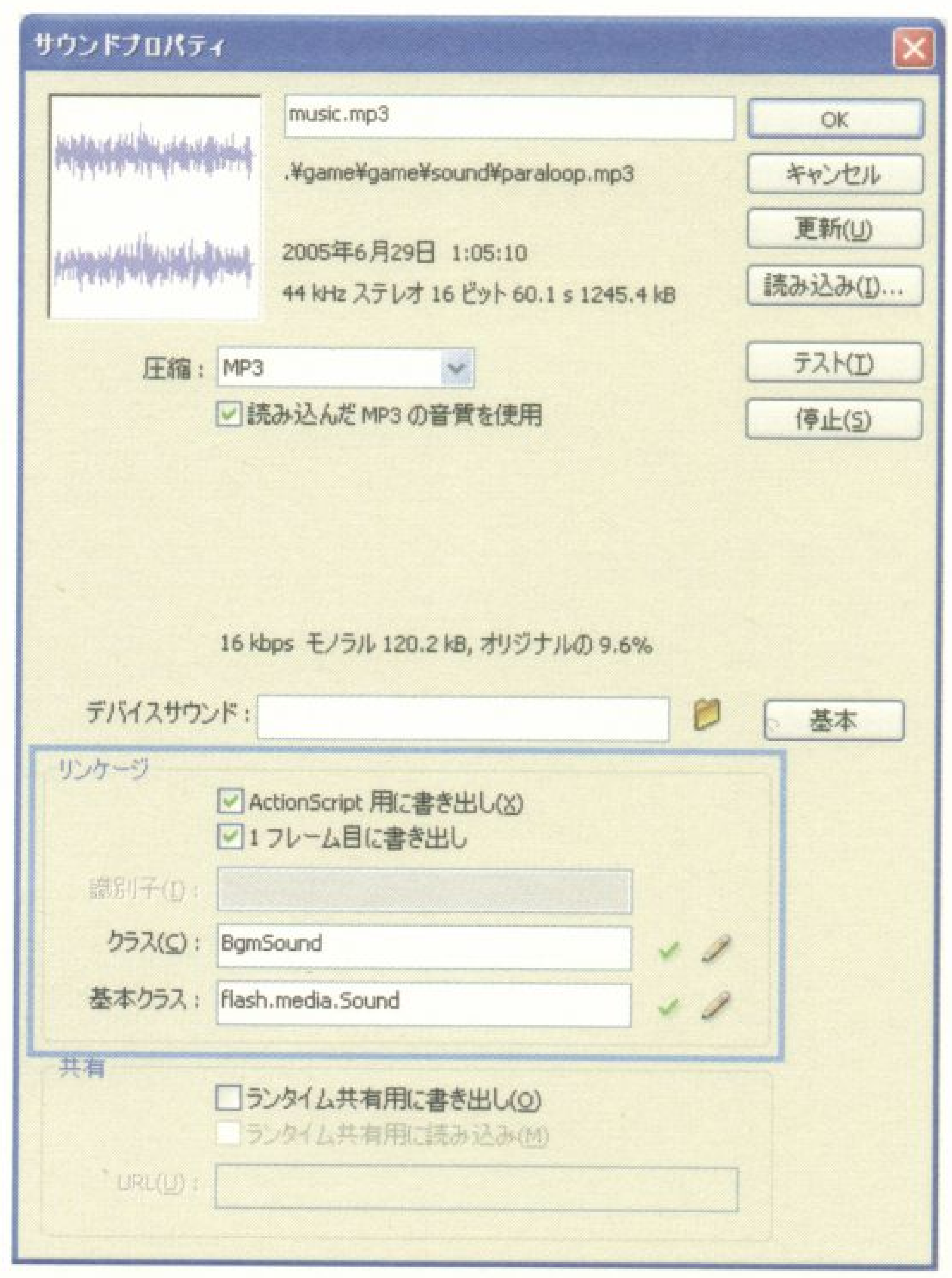
▶▶ リンケージ設定を行う

プログラムでサウンドを制御するためには、リンケージの設定を行う必要があります。これによりプログラムによるサウンドの制御が可能になります。
ライブラリ内の音声アイテムを選択し、右クリックメニューから［プロパティ...］を選びます。



［サウンドプロパティ］ウィンドウ内で、リンケージの設定を行います。
「ActionScript用に書き出し」を選択すると、リンケージに関連する各設定項目が編集可能になります。
「クラス」の項目に表示されている文字列が、プログラムでこのサウンドを呼び出す際に使用する名前となります。ここでは「BgmSound」としました。
これにより、Soundクラスを継承したクラス「BgmSound」が作られます。

■ クラス ………………………………………………… P074へ



サウンドを動的に呼び出す

リンケージ設定を行ったライブラリ内のサウンドデータを、
ActionScriptで動的に呼び出します。

▶▶ サウンドを動的に再生する

リンケージ設定を行ったサウンドは、「new リンケージで設定したクラス名( )」とすることで、呼び出すことができます。
呼び出したサウンドを再生するには、Soundインスタンスの持つメソッド「play( )」を呼びます。

```
var bgm_sound = new BgmSound();
↑ リンケージ設定したサウンド「BgmSound」をSoundインスタンスとして取得
bgm_sound.play();
```

▶▶ 再生したサウンドを停止する

再生したサウンドを制御するには、SoundChannelクラスを利用します。
再生したサウンドのSoundChannelインスタンスは、Sound.play( )の戻り値として取得することが可能です。
取得したSoundChannelインスタンスの「stop( )」メソッドを呼ぶことで、再生中のサウンドを停止することができます。

```
var bgm_sound = new BgmSound();
var sound_channel = bgm_sound.play();
↑ 戻り値からSoundChannelインスタンスを取得

sound_channel.stop(); ← 再生中のサウンドを停止
```

▶▶ 再生中のサウンドの音量を調整する

再生中のサウンドの音量の調整は、SoundChannelインスタンスのプロパティ「soundTransform」を変更することで行うことができます。
「soundTransform」プロパティにはSoundTransformインスタンスが格納されているので、自分でSoundTransformインスタンスを作成することで、音量やスピーカーのパンの設定を行うことができます。
生成時の第1引数がボリュームの設定値となり、0(無音)～1(フルボリューム)の間の数値を入れることができます。
また、生成時の第2引数はスピーカーのパンの設定値となっていて、-1(完全に左へパン)～1(完全に右へパン)の間の数値を入れることができます。
なお、0を指定するとパンが無い状態になります。

```
var bgm_sound = new BgmSound();
var sound_channel = bgm_sound.play();

sound_channel.soundTransform = new SoundTransform(0.7,1);
↑ 再生中のサウンドの音量を70%で、完全に右のスピーカーにパンしている状態
  にする
```

外部のサウンドを読み込んで再生する

Soundクラスのメソッド「Sound.load( )」を利用することで、ライブラリ内の音声ファイルだけでなく、SWFの外部にある音声データを読み込んで制御することが可能です。

▶▶ 音声ファイルを読み込んで再生する

Soundクラスのメソッド「Sound.load( )」を使うことで、Soundインスタンスに外部のmp3ファイルを読み込むことができます。読み込んだ音声データはSoundインスタンスとして、再生・音量調整などの制御をActionScriptで行うことが可能です。また、Sound.load( )を利用する際には、ファイルのURLをURLRequestインスタンスにして引数に指定する必要があります。URLRequestについては、詳しくは「5-9 Loader」で解説します。

■Loader ………………………… P158へ

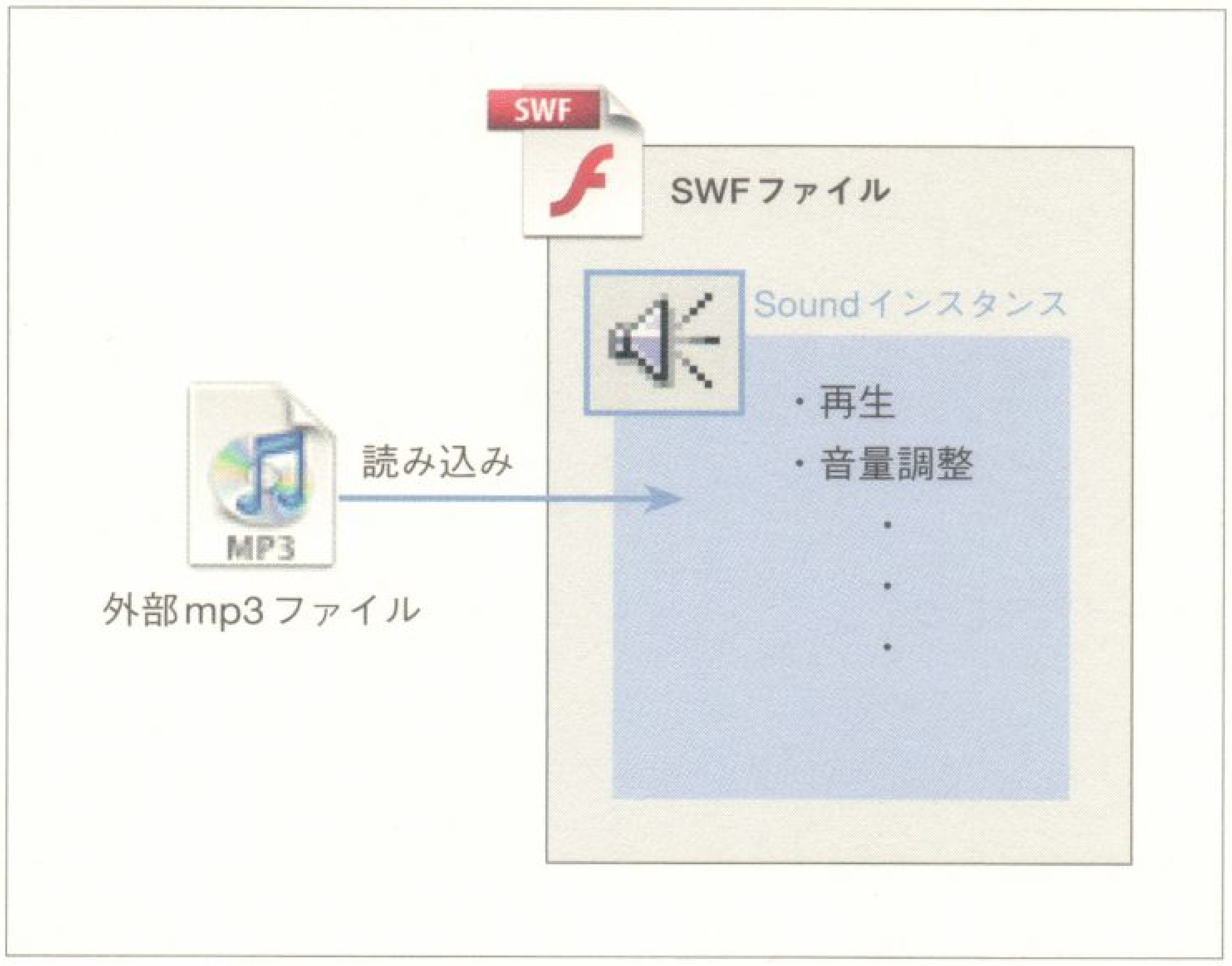
```
var sound = new Sound();// 空のSoundインスタンスを作成

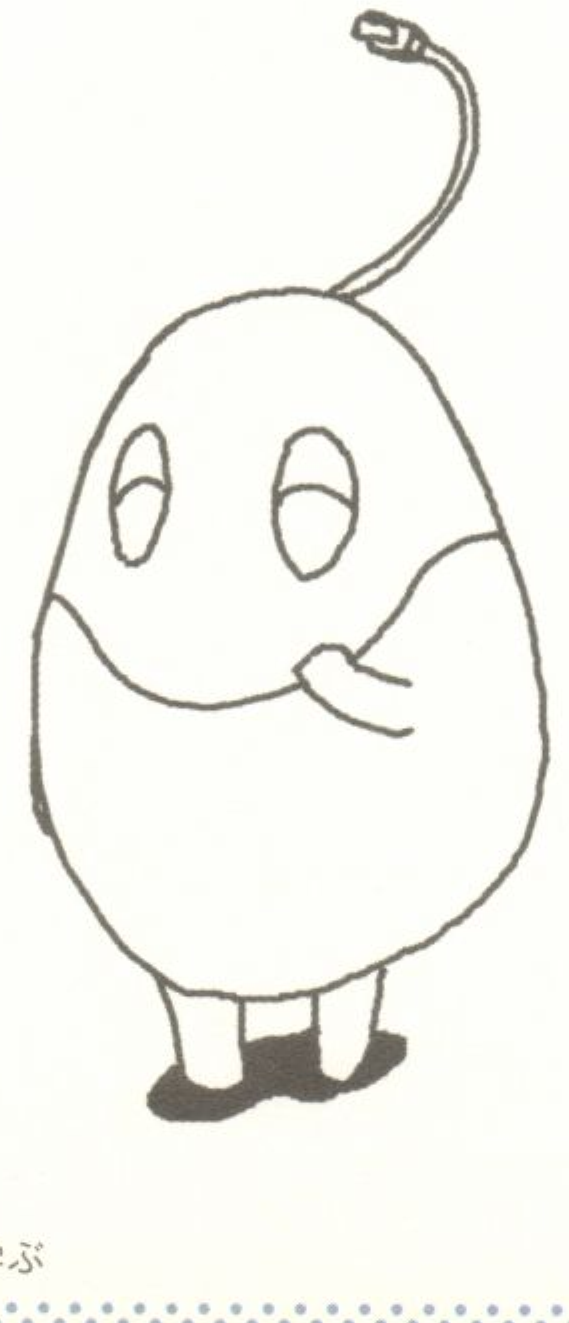
sound.addEventListener(Event.COMPLETE,onLoadSound);

var url_req = new URLRequest("sound.mp3")// ファイルのURLを指定
sound.load(url_req);// 音声ファイルの読み込み

function onLoadSound(event):void{
  sound.removeEventListener(Event.COMPLETE,onLoadSound);

  sound.play();// 読み込んだ音声データの再生
}
```



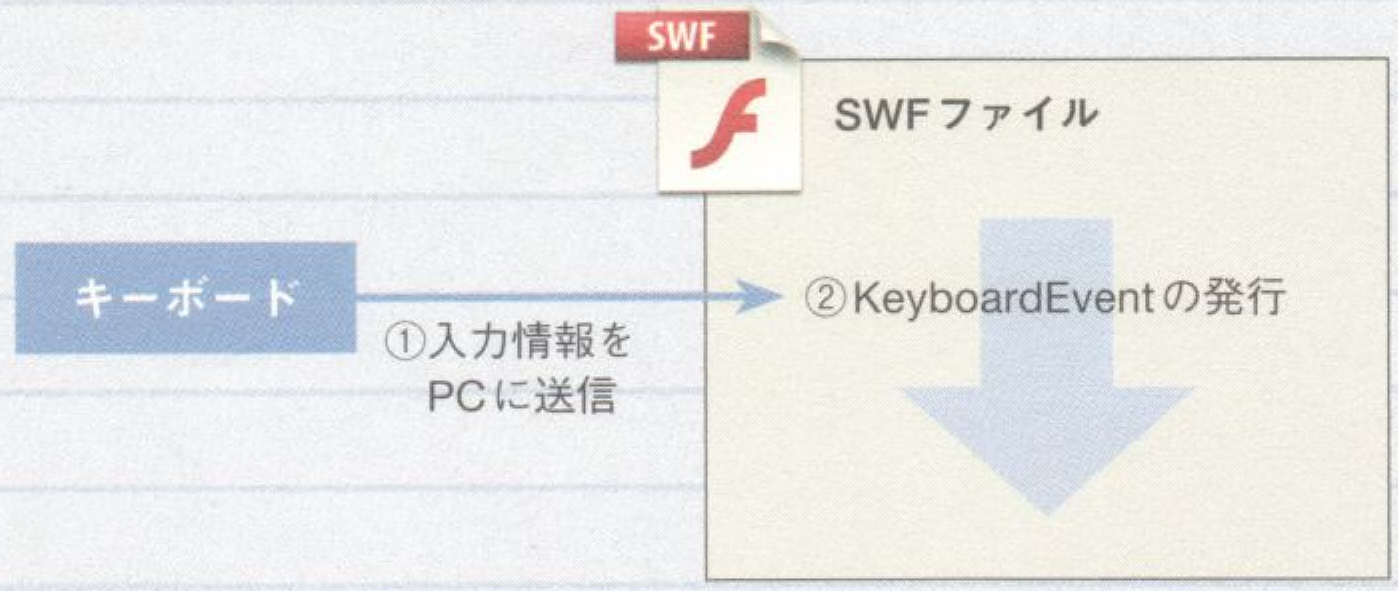


LECTURE 5-8

プログラミング［応用］

キーで操作する

Flashではマウスでの操作だけでなく、キーボードからの入力情報を受け取ることが可能です。ここでは、ActionScriptでキー入力情報を感知する方法を解説します。



KeyboardEventクラスが使われているサンプル

■ PRACTICE 8：キーボードでキャラクターを操作する ……… P278へ

合わせて読むと便利な項目

■ クラス ……… P074へ
■ イベントの概念 ……… P077へ

キーボードの入力情報を取得する

ユーザーによるキーボードへの入力は、
KeyboardEvent クラス、Keyboard クラスを用いて感知することができます。

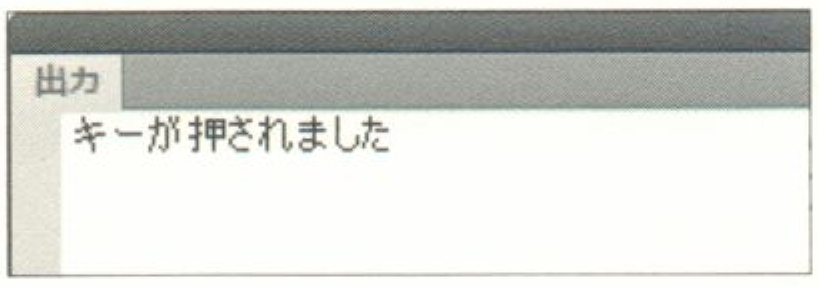
▶▶ キーボードが押されたことを感知する

キーボードが押された際の通知は、KeyboardEvent.KEY_DOWNに対してリスナーを登録することで、受け取ることが可能です。Flash上のどこにフォーカスがあたっているかに関わらず常にキーボードの状況を感知したい場合は、stageプロパティに対してリスナーを登録します。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,onKeyboardDown);

function onKeyboardDown(event){
   trace("キーが押されました");
}
```

出力結果

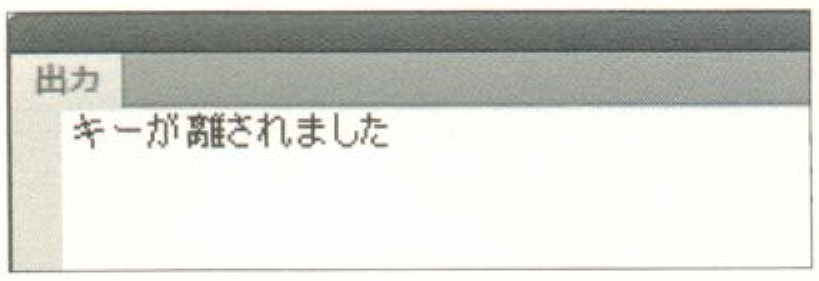


また、KeyboardEvent.KEY_UPに対してリスナーを登録することで、キーが離された際の通知を受け取ることが可能です。

```
stage.addEventListener(KeyboardEvent.KEY_UP,onKeyboardUp);

function onKeyboardUp(event){
   trace("キーが離されました");
}
```

出力結果



どのキーが押されたか判定する

KeyboardEventのプロパティ「keyCode」を用いて、キーボードのどのキーが押されたかを判定します。

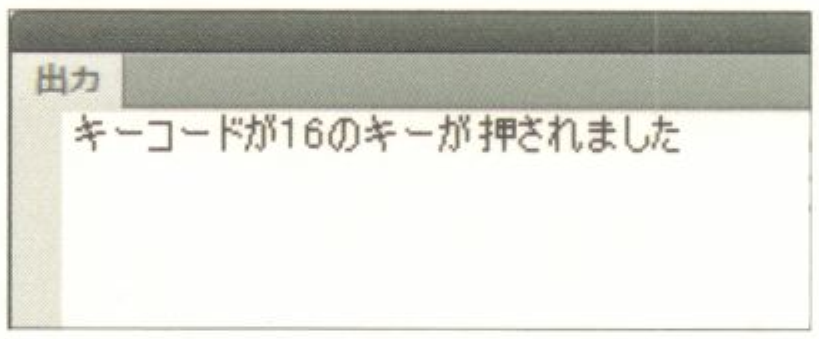
▶▶ 押されたキーのキーコードを取得する

発行されたKeyboardEventのプロパティ「keyCode」を参照することで、どのキーが押されたのか調べることが可能です。ここでは、「キーコード」と呼ばれる、キーごとに個別に割り当てられた整数が取得できます。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,onKeyboardDown);

function onKeyboardDown(event){
   var key_code = event.keyCode; //押されたキーのキーコードを取得
   trace("キーコードが" + key_code + "のキーが押されました");
}
```

出力結果

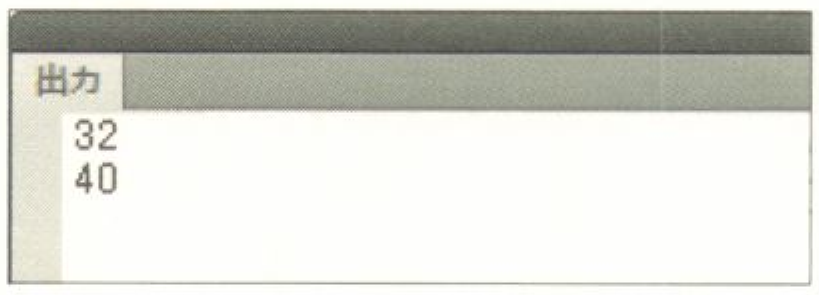


▶▶ Keyboardクラスの定数を利用する

Keyboardクラスは、主なキーのキーコードを定数として持っています。

```
trace(Keyboard.SPACE); ← スペースキーのキーコードを取得する
trace(Keyboard.DOWN); ←下方向の矢印キーのキーコードを取得する
```

出力結果



Keyboardクラスが持つ定数の主なキーコード

定数名	キー	キーコード
BACKSPACE	Backspaceキー	8
CAPS_LOCK	Caps Lockキー	20
CONTROL	Ctrlキー	17
DELETE	Deleteキー	46
DOWN	下向き矢印キー	40
END	Endキー	35
ENTER	Enterキー	13
ESCAPE	Escキー	27
HOME	Homeキー	36
INSERT	Insキー	45
LEFT	左向き矢印キー	37
NUMPAD_0	テンキーの0キー	96
NUMPAD_1	テンキーの1キー	97
NUMPAD_2	テンキーの2キー	98
NUMPAD_3	テンキーの3キー	99
NUMPAD_4	テンキーの4キー	100
NUMPAD_5	テンキーの5キー	101

定数名	キー	キーコード
NUMPAD_6	テンキーの6キー	102
NUMPAD_7	テンキーの7キー	103
NUMPAD_8	テンキーの8キー	104
NUMPAD_9	テンキーの9キー	105
NUMPAD_ADD	テンキーの加算キー	107
NUMPAD_DECIMAL	テンキーの小数点キー	110
NUMPAD_DIVIDE	テンキーの除算キー	111
NUMPAD_ENTER	テンキーのEnterキー	108
NUMPAD_MULTIPLY	テンキーの乗算キー	106
NUMPAD_SUBTRACT	テンキーの減産キー	109
PAGE_DOWN	PageDownキー	34
PAGE_UP	PageUpキー	33
RIGHT	右向き矢印キー	39
SHIFT	Shiftキー	16
SPACE	スペースバー	32
TAB	Tabキー	9
UP	上向き矢印キー	38

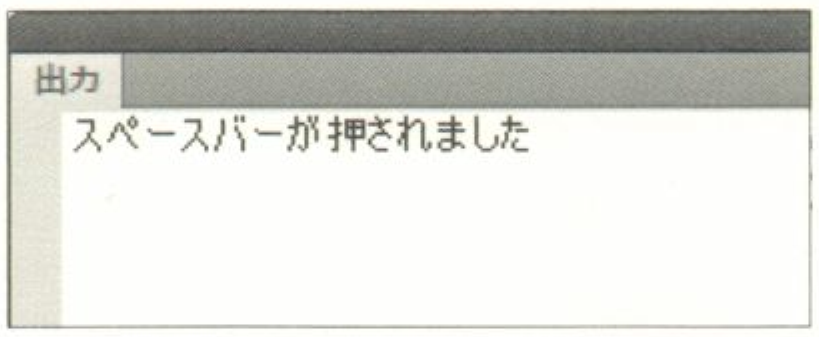
▶▶ 特定のキーが押された際の処理

if文を用いて、押されたキーのキーコードと任意のキーコードを比較することで、特定のキーが押された場合のみ実行される処理を作ることが可能です。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,onKeyboardDown);

function onKeyboardDown(event){
   var key_code = event.keyCode;
   if(key_code == Keyboard.SPACE){
    trace("スペースバーが押されました");
   }
}
```

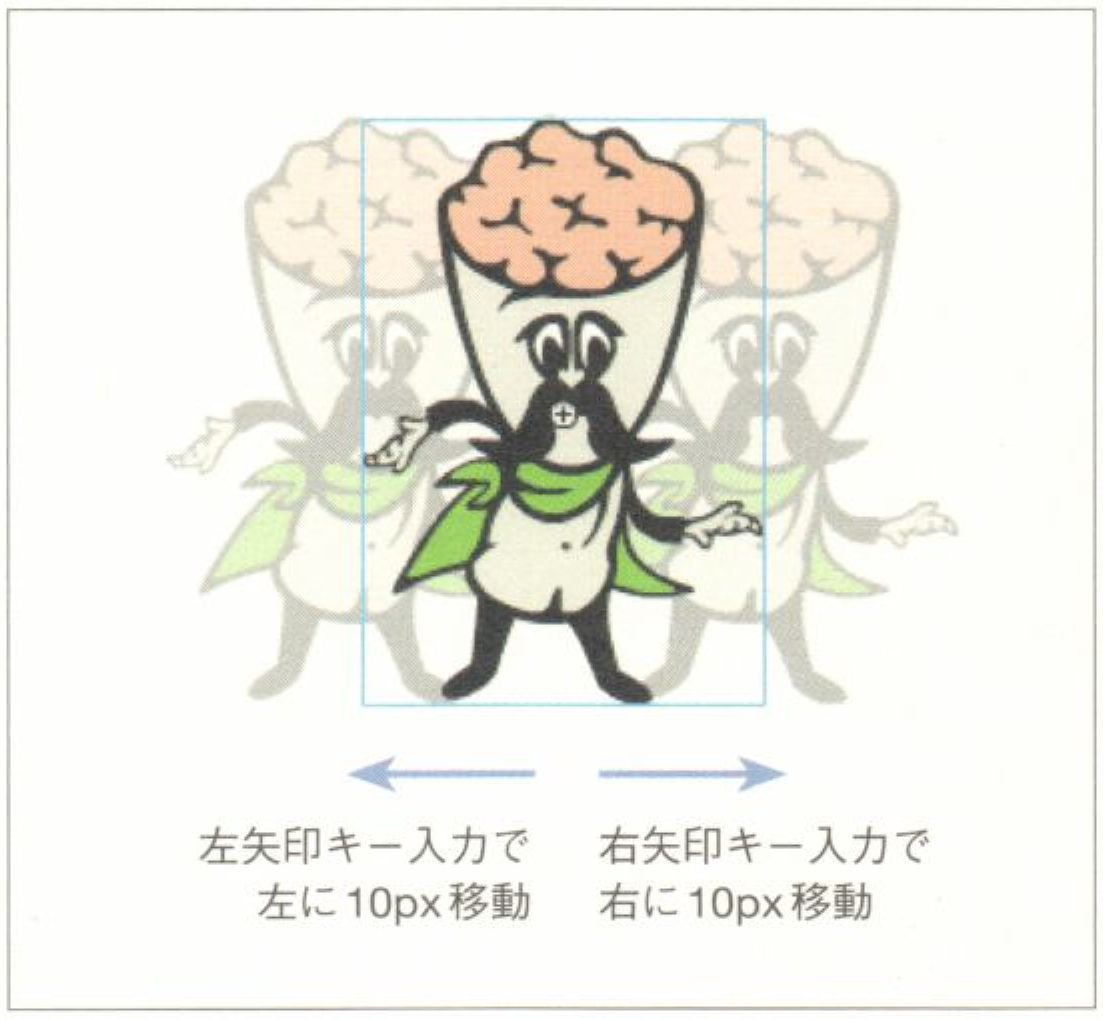
出力結果



▶▶ 左右矢印キーでムービークリップを動かす

Keyboardクラスの定数を使って、左矢印キー、右矢印キーの入力で画面上のムービークリップを動かす仕組みを作成します。

前ページの表にあるように、Keyboardクラスは、右向き矢印キーには「RIGHT」、左向き矢印キーには「LEFT」というキーコードをそれぞれ定数として持っています。これとif文による条件分岐を組み合わせることで、特定キー入力でのムービークリップの操作を実現します。




```
stage.addEventListener(KeyboardEvent.KEY_DOWN,onKeyboardDown);

function onKeyboardDown(event){
   var key_code = event.keyCode;//押されたキーのキーコードを取得

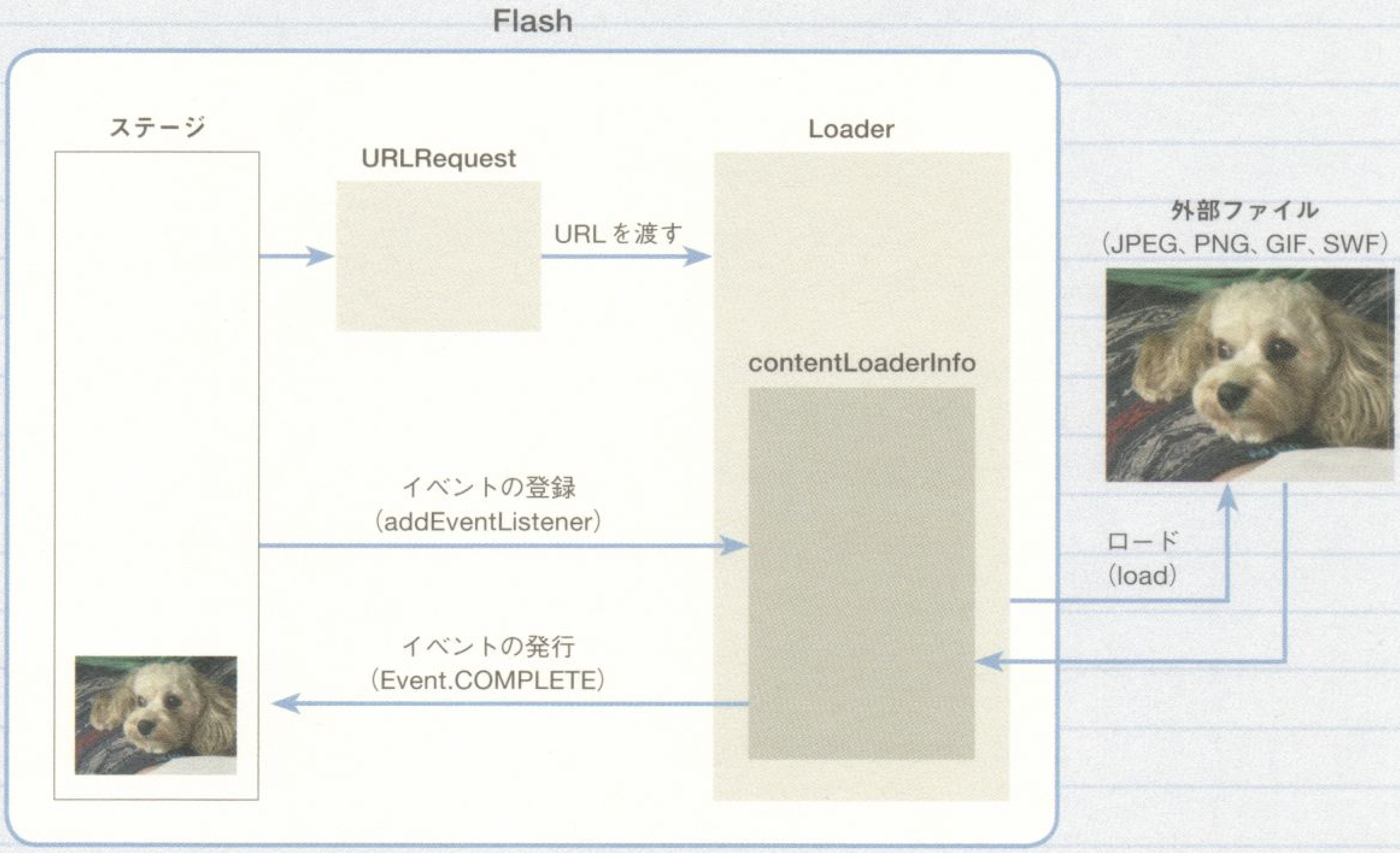
   if(key_code == Keyboard.LEFT){//キーコードが左矢印キーと同じものか判定
      chara_mc.x -= 10;
   }else if(key_code == Keyboard.RIGHT){//キーコードが右矢印キーと同じものか判定
      chara_mc.x += 10;
   }
}
```

LECTURE 5-9

プログラミング［応用］

Loader

外部イメージを読み込むにはLoaderクラスを使います。Loaderクラスが扱える外部イメージはJPEG、PNG、GIF、SWFです。Loaderクラスを使うにはその他、URLRequestクラスやイベントの概念なども必要になってきますので、合わせて勉強してください。なお、外部のXMLなどテキスト情報の読み込みはURLLoaderクラスを使いますので、そちらを参照してください。



Loaderが使われているサンプル

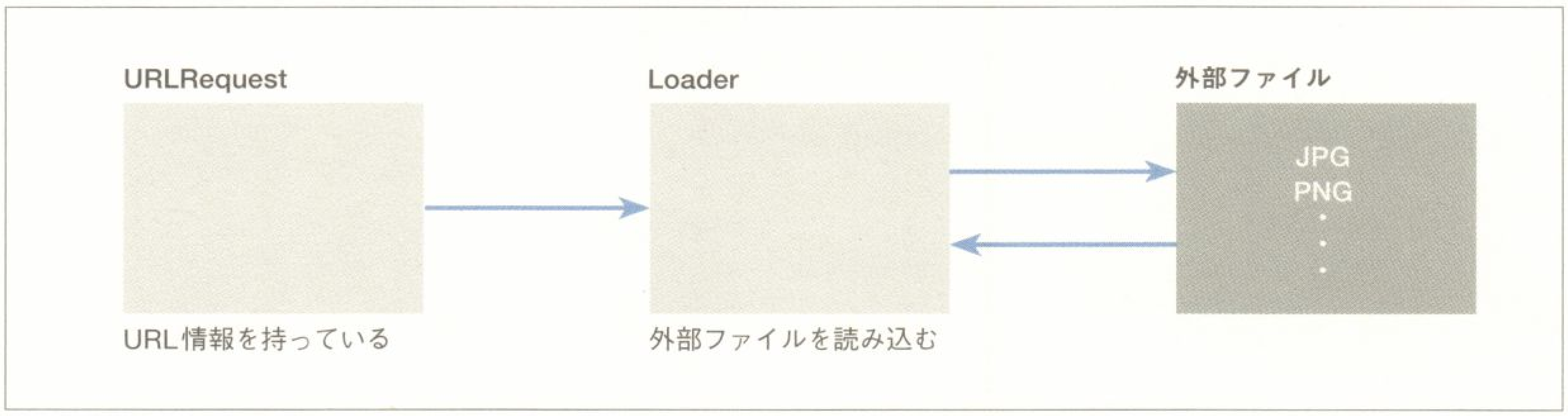
■ PRACTICE 5：
サムネイルを動的に配置する……… P224へ

合わせて読むと便利な項目

■ クラス ……………………………… P074へ

Loaderクラスの基本的な使い方

Loaderクラスは外部ファイルを読み込む機能があるのですが、外部ファイルのURL情報はURLRequestクラスが持っています。Loaderクラスを使うときは、URLRequestクラスを一緒に使います。



1. まずはLoaderクラスをインスタンス化します。

```
var myLoader = new Loader();
```

2. 次にURLRequestクラスをインスタンス化します。
   URL情報をLoaderクラスに渡すときは、かならずこのURLRequestクラスが必要になります。

```
//dog.jpgというファイルを読み込む
var myRequest = new URLRequest("dog.jpg");
```

3. Loaderクラス内のメソッドloadを呼び出します。
   このロードの際にURLRequestクラスのインスタンスを引数に渡してあげます。

```
myLoader.load(myRequest);
```

■ 引数 …………………………………………………… P088へ

4. 最後にLoaderクラスを画面に配置します。これをしないと画面上に表示されません。

```
addChild(myLoader);
```

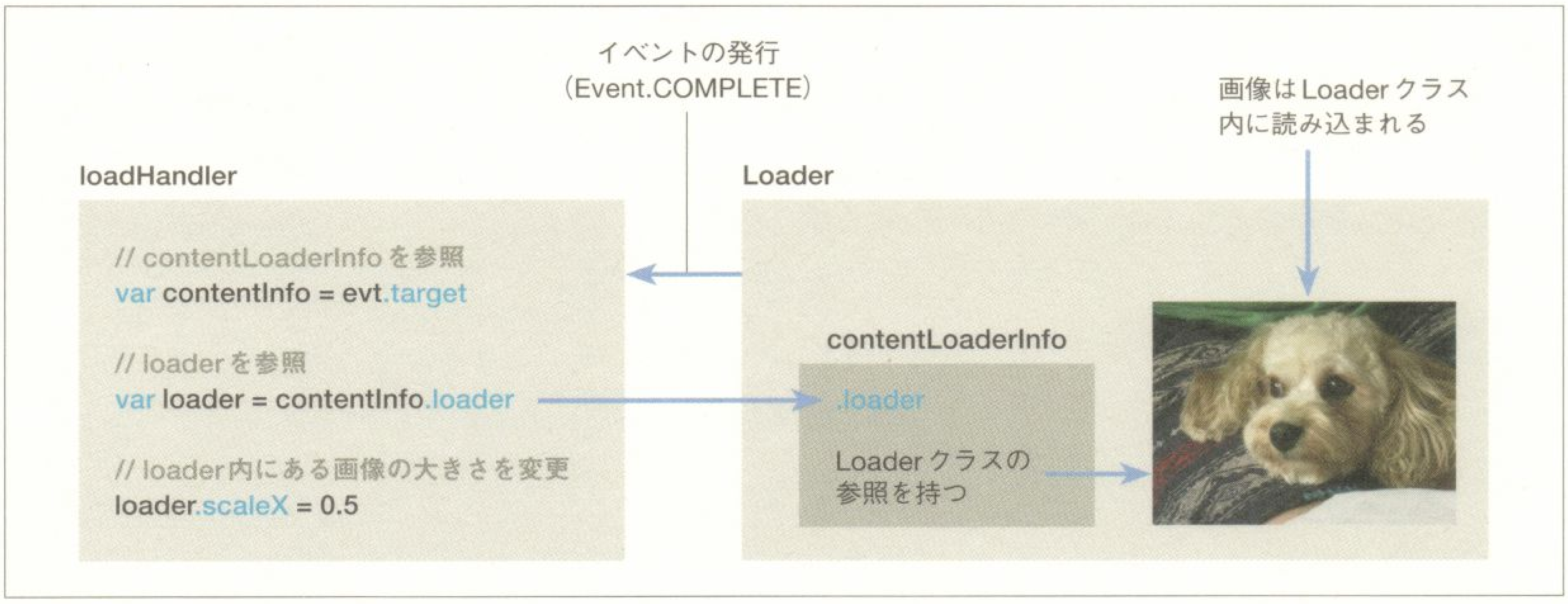
■ addChild …………………………………………… P124へ

5. すべてのコードを書くとこのようになります。

```
//Loaderクラスのインスタンス化
var myLoader = new Loader();
//URLRequestクラスのインスタンス化
var myRequest = new URLRequest("dog.jpg");
//外部ファイルの読み込み開始
myLoader.load(myRequest);
//画面に配置
addChild(myLoader);
```

イメージ読み込み完了イベントを受け取る

前ページのやり方でもイメージを読み込むことができます。
ただし、イメージがいつ読み込み終わったのかのタイミングを知ることはできません。
画像の大きさを変更するにはファイルの読み込みが終わってから行う必要があるので、その処理を紹介します。



1. 前ページのコードにEvent.COMPLETEのイベントを受け取る処理を追加します。ただしここではLoaderクラス内にあるcontentLoaderInfoプロパティに対してaddEventListenerを行います。
   contentLoaderInfoはファイル容量や幅、高さなどの情報を含んでいます。

```
var myLoader = new Loader();
var myRequest = new URLRequest("dog.jpg");
myLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, loadHandler);
⬆ イメージがロード完了したときのイベントを登録
myLoader.load(myRequest);
addChild(myLoader);
```

■ イベントの概念 ……………………………… P077へ

2. 手順1で設定したイベントを受け取る処理を書きます。

```
function loadHandler(evt) {

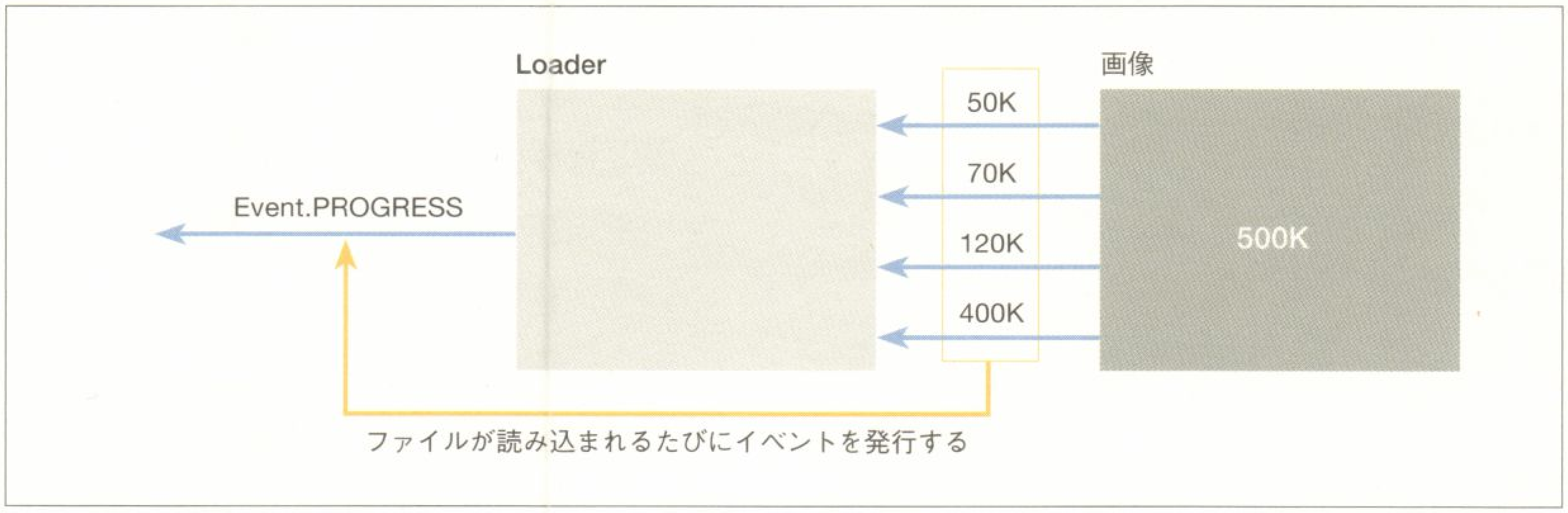
}
```

3. loadHanlder内に画像の大きさを変更する処理を書きます。evt引数の中にはtargetというプロパティが含まれており、そこにはcontentLoaderInfoの参照が含まれています。

```
function loadHandler(evt) {
   var loaderInfo = evt.target; ⬅ evt.targetがcontentLoaderInfoを指す
   var loader = loaderInfo.loader;
   ⬆ contentLoaderInfo内にはloaderプロパティがあり、これが画像の入っているLoaderのことを指す
   loader.scaleX = 0.5; ⬅ 画像に対して大きさを変更できる
   loader.scaleY = 0.5;
}
```

プリローダーを作成する

contentLoaderInfoではイメージのロード完了イベント以外に、現在読み込まれているファイルサイズを知ることができます。イメージ読み込み完了にはEventクラスを使いましたが、今回はProgressEventクラスを使用します。



1. contentLoaderInfoにProgressEvent.PROGRESSというイベントを登録します。

```
var myLoader = new Loader();
var myRequest = new URLRequest("dog.jpg");
myLoader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progressHandler);
  ⬆ ProgressEvent.PROGRESSイベントはファイルが少しずつ読み込まれるたびに呼び出される
myLoader.load(myRequest);
addChild(myLoader);
```

2. 受け取る側の引数evt内にbytesLoadedとbytesTotalというプロパティがあります。これらをもとに何パーセント読み込まれているかを計算します。

```
function progressHandler(evt){
    // パーセント = 現在読み込まれているファイルサイズ ÷ すべてのファイルサイズ × 100
  var percent = evt.bytesLoaded / evt.bytesTotal * 100;
    trace(percent + "読み込まれました。")
}
```

3. すべてのコードを合わせるとこのようになります。

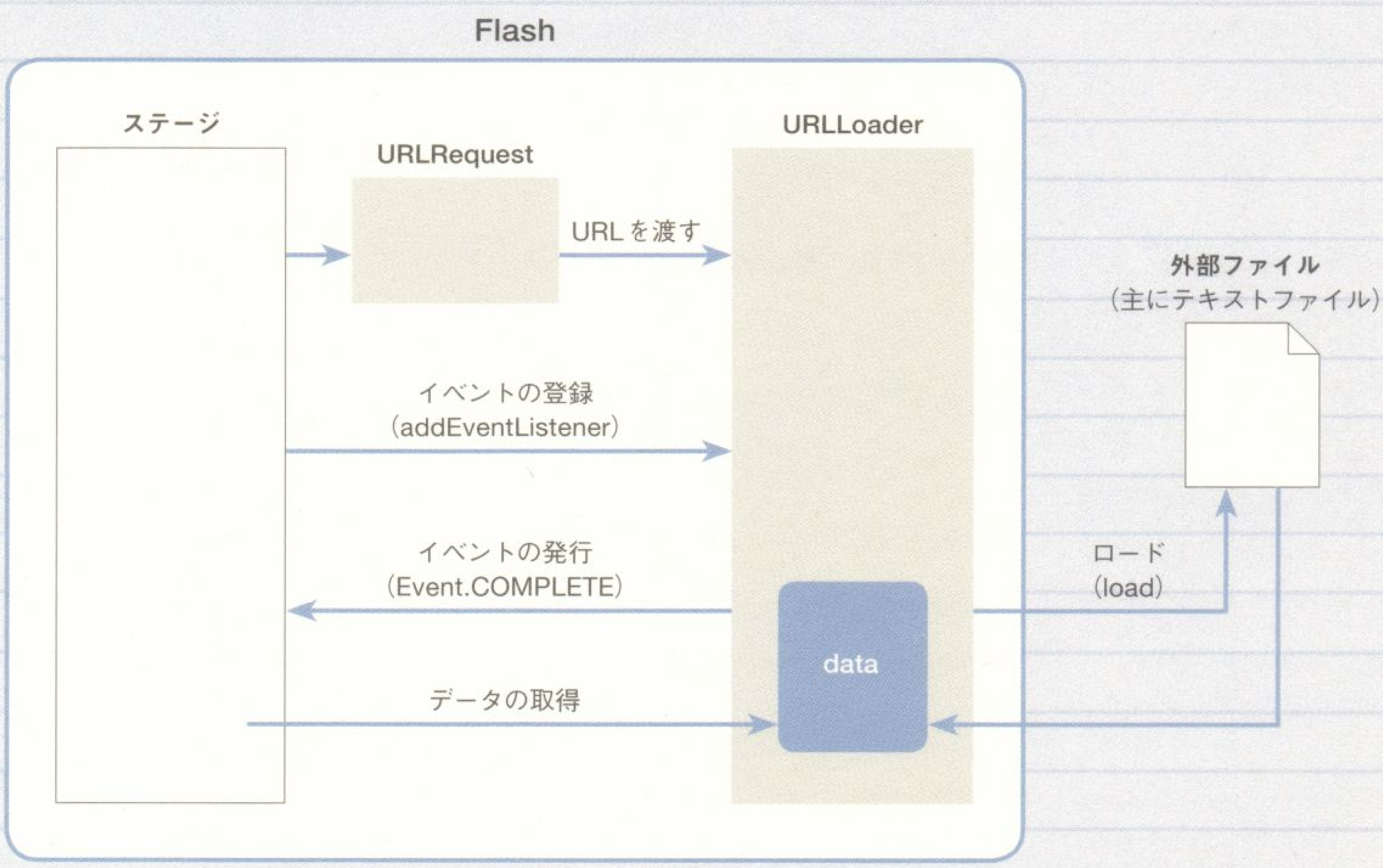
```
var myLoader = new Loader();
var myRequest = new URLRequest("dog.jpg");
myLoader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myLoader.load(myRequest);
addChild(myLoader);

function progressHandler(evt){
    var percent = evt.bytesLoaded / evt.bytesTotal * 100;
    trace(percent + "読み込まれました。")
};
```

LECTURE 5-10

プログラミング［応用］ URLLoader

外部データを読み込むにはURLLoaderクラスを使います。URLLoaderはアクセスできる外部ファイルであれば、どのような種類のファイルも読み込みできますが、通常はテキストファイルやXMLファイルなどの文字情報の読み込みに利用します。URLLoaderクラスを使うにはその他、URLRequestクラスやイベントの概念なども必要になってきますので、合わせて勉強してください。なお、SWFや画像ファイルを読み込むにはLoaderクラスを使うほうが一般的ですので、そちらを参照してください。



URLLoaderが使われているサンプル

■ PRACTICE 6：
RSS情報を読み込む ……………… P238へ

合わせて読むと便利な項目

■ イベントの概念 …………………… P077へ

URLLoaderの基本的な使い方

先ほども書きましたが、URLLoaderを使うにはURLRequestクラスなどと一緒に使う必要があります。
使い方の流れをきちんと把握してください。

1. まずはURLLoaderクラスを初期化し、URLLoaderのインスタンスを生成します。

```
var urlLoader = new URLLoader();
```

2. 読み込むデータの種類を指定します。
テキスト情報を読み込むURLLoaderDataFormat.TEXTのほか、バイナリデータなどを読み込むためのURLLoaderDataFormat.BINARYとURLエンコード形式の変数を読み込むためのURLLoaderDataFormat.VARIABLESを指定可能です。

```
urlLoader.dataFormat = URLLoaderDataFormat.TEXT;
```

3. 次に、URLRequestクラスをインスタンス化し、URLRequestのインスタンスを生成します。
URL情報をURLLoaderに渡すときは、必ずこのURLRequestインスタンスが必要になります。

```
var req = new URLRequest("data.xml");
```

4. URLLoaderインスタンスのメソッドloadを呼び出します。ロードのときにURLRequestインスタンスを引数に渡してあげます。

```
urlLoader.load(req);
```

■引数 P088へ

5. すべてのコードを書くと右のようになります。

```
var urlLoader = new URLLoader();
urlLoader.dataFormat = URLLoaderDataFormat.TEXT;
var req = new URLRequest("data.xml");
urlLoader.load(req);
```

読み込み完了イベントを受け取る

前ページの方法でデータを読み込むまではできますが、データがいつ読み込み終わったのかのタイミングを知ることはできません。データの内容を取得するには読み込み終わってからでないとできないので、この読み込み終わりのタイミングを受け取る処理を紹介します。

1. 前ページのコードにEvent.COMPLETEのイベントを受け取る処理を追加します。
   URLLoaderインスタンスに対して、addEventListenerを行います。

```
urlLoader.addEventListener(Event.COMPLETE, urlLoader_completeHandler);
```

■イベントの概念 ································ P077へ
■イベントリスナ ································ P090へ

2. 手順1で設定したイベントを受け取る処理を書きます。

```
function xmlLoader_completeHandler(evt)
{

}
```

3. urlLoader_completeHandler内に、ダウンロードしたデータを取得する処理を書きます。
   evt引数の中にはtargetというプロパティが含まれており、そこにはURLLoaderインスタンスの参照が含まれています。そして、ダウンロードして取得したデータはURLLoaderインスタンスのdataプロパティに格納されています。

```
evt.target
```

4. すべてのコードを合わせるとこのようになります。

```
var urlLoader = new URLLoader();
urlLoader.dataFormat = URLLoaderDataFormat.TEXT;
urlLoader.addEventListener(Event.COMPLETE, urlLoader_completeHandler);
var req = new URLRequest("data.xml");
urlLoader.load(req);

function urlLoader_completeHandler(evt)
{
    var textData = evt.target.data;
}
```

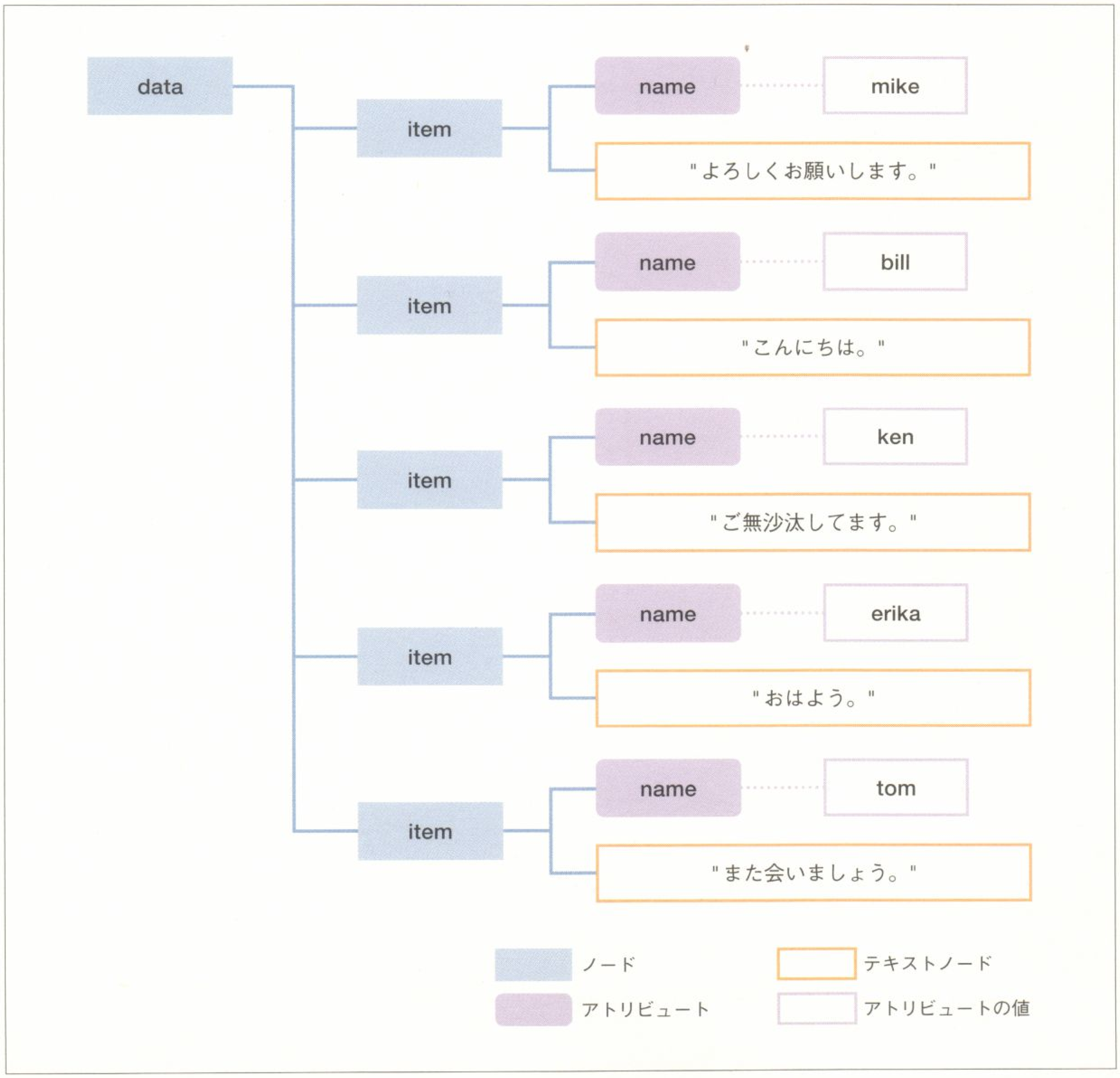
XMLを解析する

読み込んだテキストデータについて、
読み込み完了後に必要なデータを取得するための処理を行います。

ダウンロードしたデータがXMLの場合は、それを使ってXMLオブジェクトを作成し、XML情報を解析して必要なデータを取り出すことになります。
なお、読み込むXMLデータは右のようなものとし、その中でitemノードのname属性と、itemノードの子テキストノードのデータを抽出することにします。

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <item name="mike"><![CDATA[よろしくお願いします。]]></item>
  <item name="bill"><![CDATA[こんにちは。]]></item>
  <item name="ken"><![CDATA[ご無沙汰してます。]]></item>
  <item name="erika"><![CDATA[おはよう。]]></item>
  <item name="tom"><![CDATA[また会いましょう。]]></item>
</data>
```

XML構造を図示すると、次のようになります。



LECTURE 5-10

1. 読み込んだテキストデータを引数として XMLオブジェクトを作成します。

```
var xml = new XML(textData);
```

2. itemノードのname属性をすべて取得する場合の処理を記述します。

```
for each(var itemXml in xml.item)
{
    var name = itemXml.@name;
}
```

3. itemノードの子テキストノードをすべて取得する場合の処理を記述します。

```
for each(var itemXml in xml.item)
{
    var message = itemXml.text();
}
```

4. すべてのコードを合わせるとこのようになります。

```
function urlLoader_completeHandler(evt)
{
    var textData = evt.target.data;

    var xml = new XML(textData);

    for each(var itemXml in xml.item)
    {
        trace(itemXml.@name);
        trace(itemXml.text);
    }
}
```

出力結果

```
出力
名前：mike
メッセージ：よろしくお願いします。
名前：bill
メッセージ：こんにちは。
名前：ken
メッセージ：ご無沙汰してます。
名前：erika
メッセージ：おはよう。
名前：tom
メッセージ：また会いましょう。
```

LECTURE 5-11

プログラミング［応用］

navigateToURL()

Flashからブラウザのウインドウを新しく開いてHTMLページを表示させる、または同じページ内にHTMLを置き換えるときに使うのが、navigateToURL( )メソッドです。リンクを作るようなイメージなので、Flashバナーを作る際にも使用できます。ここでは、このnavigateToURL( )メソッドの使い方を見ていきましょう。

navigateToURL()が使われているサンプル

■ PRACTICE 2：HTMLページを切り替える仕組みを作る ……………………………………………P199へ

navigateToURL( )メソッドの使い方

navigateToURL( )メソッドは、第1引数に移動先のURLがプロパティとして設定されているURLRequestクラスのオブジェクト、第2引数にウインドウ名を指定します。

使い方

```
navigateToURL(URLRequestオブジェクト, ウインドウ名);
```

※移動先のURLや、ヘッダ名、データ送信方法などは、URLRequestオブジェクトのプロパティで設定します。

```
btn_mc.addEventListener(MouseEvent.CLICK, openNewWindow); ⇐ イベントリスナの登録

function openNewWindow(event){
  var myRequest:URLRequest = new URLRequest(); ⇐ URLRequestオブジェクトを作成
  myRequest.url = "http://www.cshool.jp/"; ⇐ URLRequestのプロパティurlに移動先のURLを代入
  navigateToURL(myRequest, "_blank"); ⇐ ブラウザの新規ページを開き、URLのページに移動
}
```

ウインドウ名の部分は、フレーム名でもよいですし、上のサンプルのように、"_blank"、"_self"、"_parent"、"_top"という形でも指定できます。ウインドウ名を指定しない場合は、"_blank"がデフォルト値となります。

_self	現在のウィンドウ内の現在のフレーム
_blank	新規ウィンドウ
_parent	現在のフレームの親
_top	現在のウィンドウ内の最上位のフレーム

▶▶ navigateToURL( )メソッドの応用

navigateToURL( )メソッドは、他にもメール作成画面を開いたり、またCGIなどにHTTP/HTTPSリクエストをする際のメソッドを指定することもできます。

メール作成画面を開く

メール作成画面を開くには、URLRequestのurlプロパティに、mailto:(メールアドレス)を入れます。

```
btn_mc.addEventListener(MouseEvent.CLICK, createNewMail); ⬅ イベントリスナの登録

function createNewMail(event){
    var myRequest:URLRequest = new URLRequest(); ⬅ URLRequestオブジェクトを作成
    myRequest.url = "mailto:test@test.com"; ⬅ URLRequestのプロパティurlにmailtoを代入
    navigateToURL(myRequest); //メール作成画面が表示される
}
```

GET/POSTメソッドを使用したデータの送信

フォームのデータなどをGET/POSTメソッドを使って送信するときは、データを格納するURLVariablesオブジェクトを作成し、そのデータをURLRequestのプロパティdataに渡します。メソッドはmethodを使います。

```
btn_mc.addEventListener(MouseEvent.CLICK, openNewWindow); ⬅ イベントリスナの登録

function openNewWindow(event){
    var myData:URLVariables = new URLVariables(); ⬅ 送信データ用のURLVariablesオブジェクトを作成
    myData.age = 30;
    myData.tel = "090-XXX-XXXX";

    var myRequest:URLRequest = new URLRequest; ⬅ URLRequestオブジェクトを作成
    myRequest.url = "http://www.hoge.com/form.cgi/"; ⬅ URLRequestのプロパティurlに移動先のURLを代入
    myRequest.method = URLRequestMethod.POST; ⬅ URLRequestのプロパティmethodで、送信方法をPOSTに指定
    myRequest.data = myData; ⬅ URLRequestのプロパティdataに、送信するデータを代入
    navigateToURL(myRequest);
}
```

▶▶ URLRequestオブジェクトのプロパティ

URLRequestオブジェクトのプロパティには以下のものが用意されています。

URLRequestオブジェクトのプロパティ

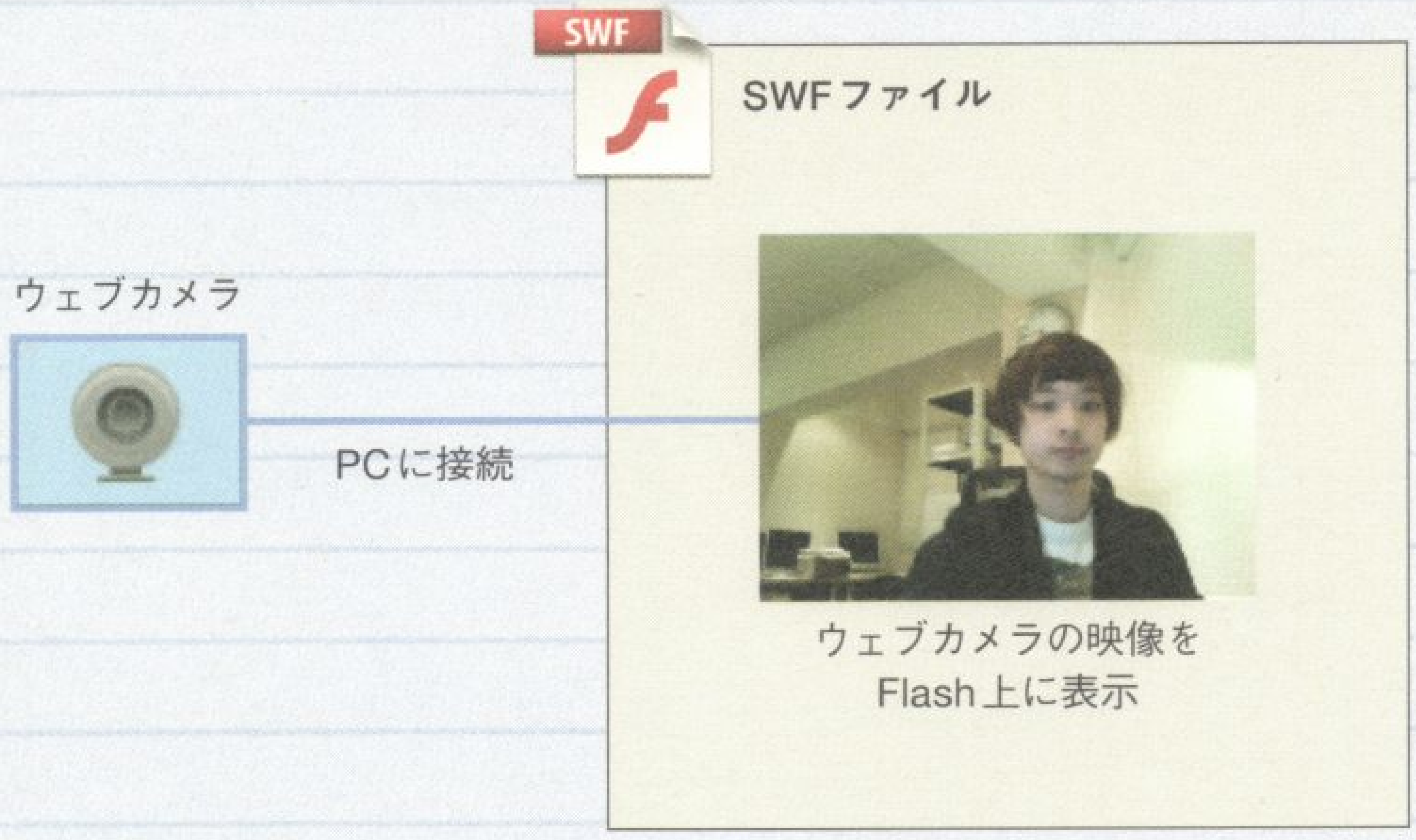
プロパティ	データ型	説明
contentType	String	data プロパティのコンテンツの MIME コンテンツタイプ
data	Object	URL リクエストで送信されるデータを含むオブジェクト
digest	String	署名付き Adobe プラットフォームコンポーネントを一意に識別するストリング
method	String	HTTP フォーム送信メソッド
requestHeaders	Array	HTTP リクエストヘッダの配列を HTTP リクエストに追加
url	String	リクエストされるURL

LECTURE 5-12

プログラミング［応用］

ウェブカメラ

Flashでは、コンピュータに接続されたウェブカメラの制御や、ウェブカメラで写した映像を画面に表示することが可能です。ActionScriptでカメラを制御するにはCameraクラスを利用します。



Cameraクラスが使われているサンプル

- PRACTICE 8：
  ウェブカメラの起動と制御………… P274へ

合わせて読むと便利な項目

- addChild ……………………………… P124へ
- イベントの概念 ……………………… P077へ

カメラをActionScriptで制御する

Cameraクラスを用いて、コンピュータに接続されたカメラの取得やカメラで写した映像を画面に表示する方法を解説します。

▶▶ Cameraインスタンスを取得する

PCに接続されたカメラを取得・制御するには、Cameraクラスを使用します。
Camera.getCamera( )を実行することで、Cameraインスタンスを取得できます。また、任意のCameraインスタンスを取得したい場合は、引数にCameraインスタンスの名前を指定します。

```
//Cameraインスタンスを取得
var camera = Camera.getCamera();
```

LECTURE 5-12

▶▶ 取得したCameraインスタンスが利用可能かどうか調べる

Cameraインスタンスからのカメラへのアクセスにはユーザーの許可が必要となります。
取得したCameraインスタンスが利用可能かどうかはmutedプロパティを参照することで、調べることが可能です。
mutedプロパティがtrueの場合はそのCameraインスタンスは利用することができず、falseの場合は利用可能となります。

```
//Cameraインスタンスを取得
var camera = Camera.getCamera();

//Cameraインスタンスが利用可能か判定
if(camera.muted == true){
   trace("取得したCameraインスタンスは利用できません");
}else{
   trace("取得したCameraインスタンスは利用可能です");
}
```

▶▶ ユーザーにCameraインスタンスへのアクセスを許可させる

Cameraインスタンスへアクセスするためには、ユーザーが［Flash Playerの設定］ダイアログボックスで許可の設定を行う必要があります。
Security.showSettings(SecurityPanel.PRIVACY)を呼び出すことで、Flash Player上にダイアログを開くことができます。

```
//Cameraインスタンスを取得
var camera = Camera.getCamera();
//Cameraインスタンスが利用可能か判定
if(camera.muted == true){
   camera.addEventListener(StatusEvent.STATUS, onStatus);
   //設定ダイアログを開く
   Security.showSettings(SecurityPanel.PRIVACY);
}else{
   trace("取得したCameraインスタンスは利用可能です");
}

function onStatus(event){
   //Cameraインスタンスが利用可能か判定

   if (camera.muted == false) {
      camera.removeEventListener(StatusEvent.STATUS, onStatus);
      trace("取得したCameraインスタンスが利用可能になりました");
   }
}
```

また設定が変更された場合は、Cameraインスタンスの発行するStatusEvent.STATUSに対してリスナーを登録しておくことで、イベントの通知を受け取ることが可能です。



▶▶ 画面上にカメラを表示する

カメラに写っている映像を画面上に表示するためには、CameraインスタンスをVideoインスタンスに関連付ける必要があります。
表示したいCameraインスタンスを引数に指定して、video.attachCamera( )を呼び出すことで、CameraインスタンスとVideoインスタンスを関連付けることができます。関連付けられたVideoインスタンスは画面上にaddChild( )することで、表示が可能です。

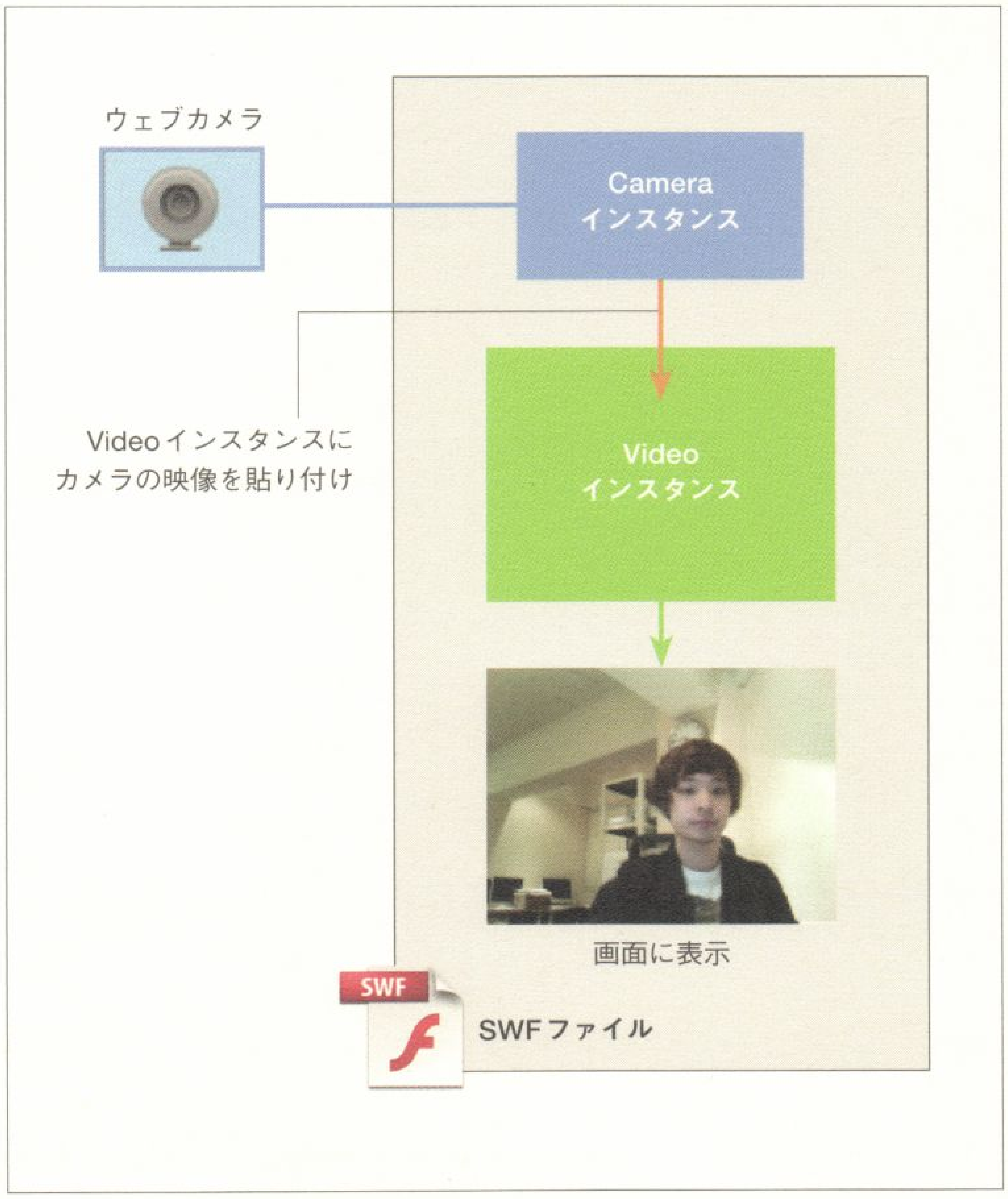
```
//Cameraインスタンスを取得
var camera = Camera.getCamera();

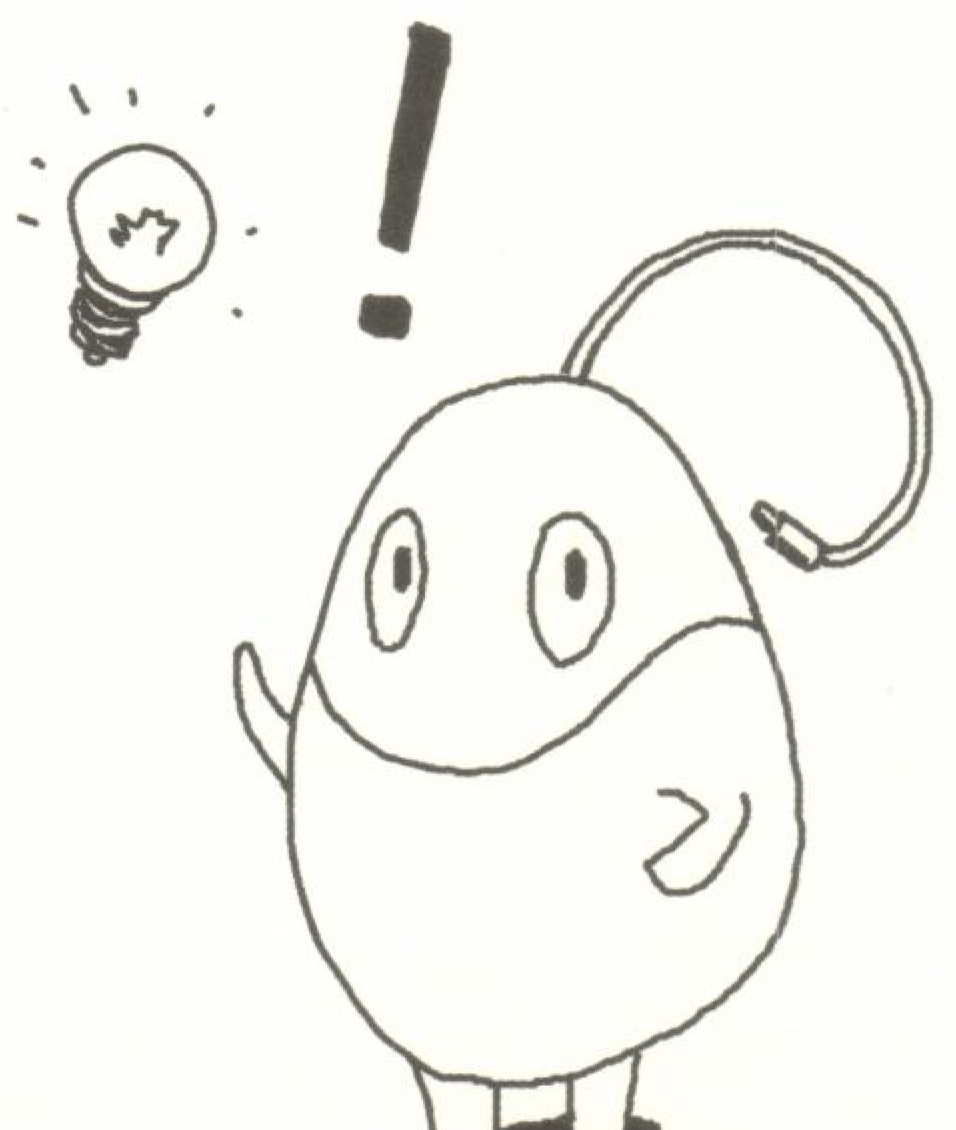
//Cameraインスタンスが利用可能か判定
if(camera.muted == true){
  camera.addEventListener(StatusEvent.STATUS,onStatus);
  //ダイアログボックスを開く
  Security.showSettings(SecurityPanel.PRIVACY);
}else{
  showVideo();
}


function onStatus(event){
  if(camera.muted == false){
    camera.removeEventListener(StatusEvent.STATUS,onStatus);
    showVideo();
  }
}

function showVideo(){
  //カメラのサイズに合わせたVideoインスタンスを作成
  var video = new Video(camera.width,camera.height);
  //Videoインスタンスにカメラの映像を貼り付ける
  video.attachCamera(camera);

  //Videoインスタンスを画面に表示
  addChild(video);
}
```





PART 3

Flashで作る

このPARTでは実際のサンプルを紹介しながら技術の解説をしていきます。
まずは自分の作りたいものと近いものを探し、そこから始めてみてください。
個別の細かな技術説明はPART 2に書いていますので、そちらと行き来しながら勉強すると良いでしょう。また、サンプルがダウンロードできますので、サンプルを開きながら読んでみてください。

PRACTICE 1

タイムラインアニメーションを作る

表現力：★★★★☆　プログラミング力：★☆☆☆☆

バナー制作など細かい動きのある表現には、タイムライン上でアニメーションさせるのが一般的です。このサンプルではタイムラインで動くアニメーションを作成し、Flashアニメーションの基本を習得していきます。シンボルをタイムライン上に入れ子構造にしたものを配置して、各パーツを動かしていきます。モーショントゥイーンを使用するとともに、イージングを使用した、より効果的なアニメーション表現を学んでいきます。今回はマンガのコマの枠を例に挙げて、マスクについても触れていきます。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

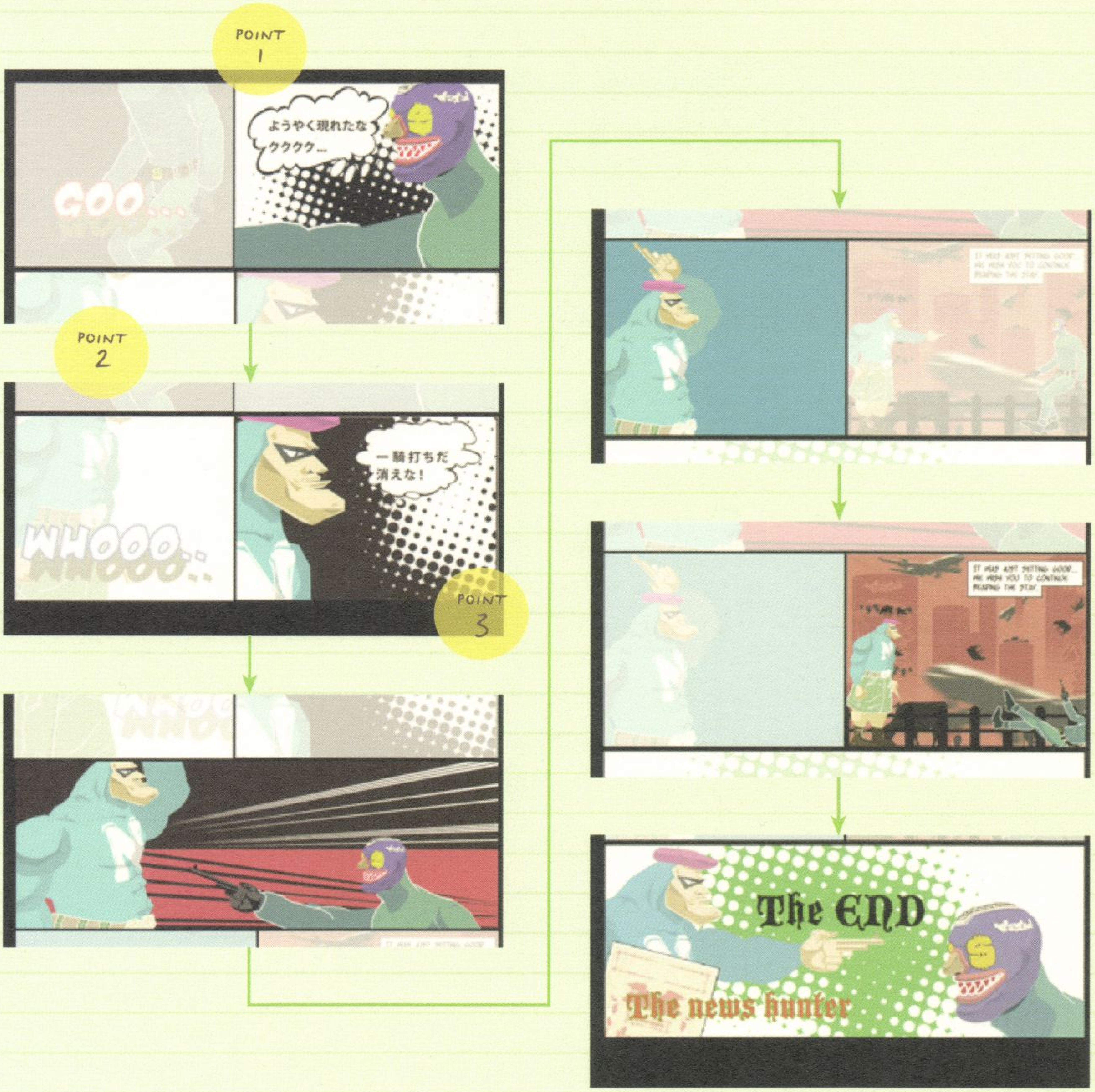
- フレームレート……………………………… P027へ
- タイムライン ……………………………… P038へ
- シンボル ……………………………… P034へ
- アニメーションを作る……………………………… P046へ
- マスクを使う ……………………………… P051へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- シンボルの階層構造
- ボーンツールの使用

サンプルの説明



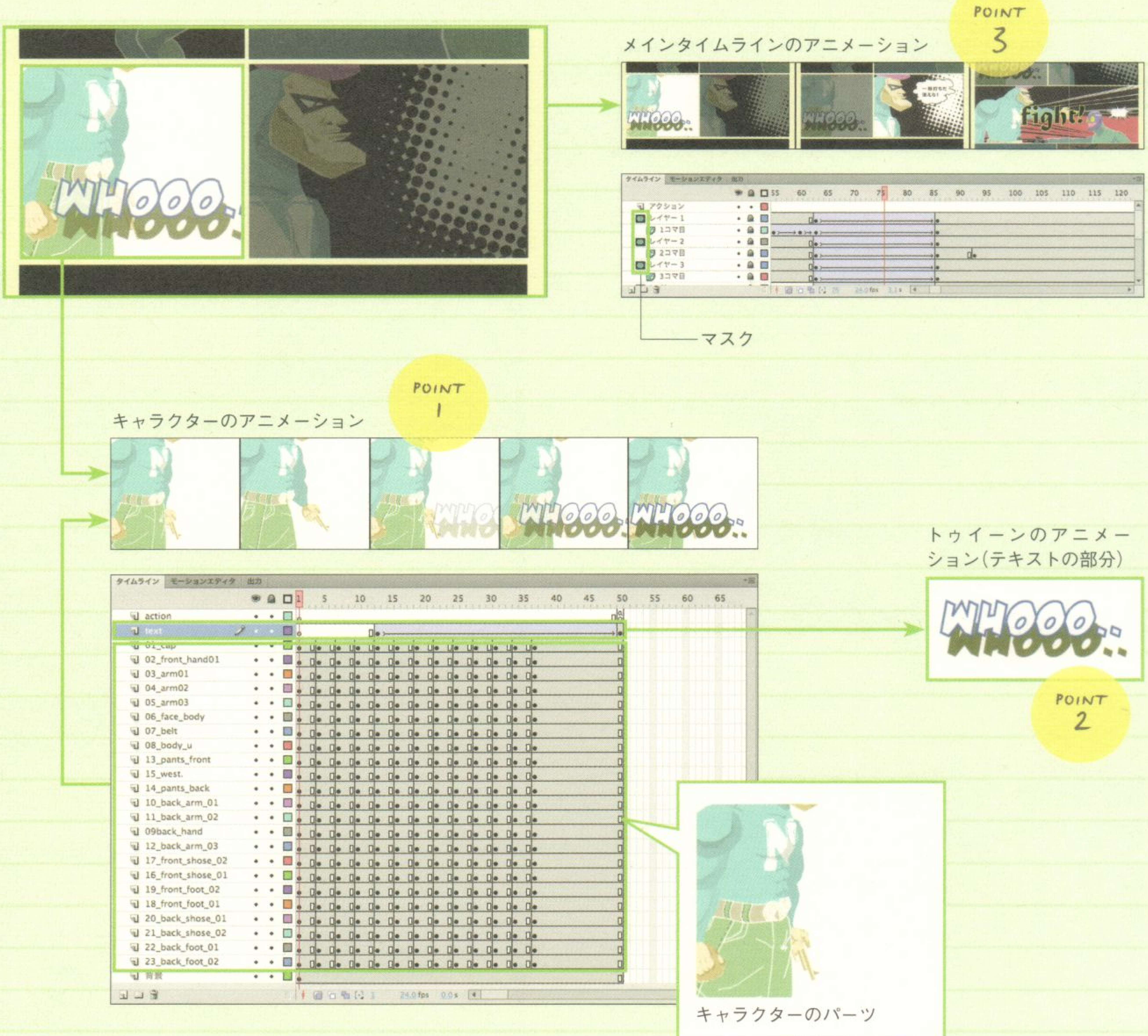
POINT

1 マンガのコマごとに枠の中身がアニメーションします。現在再生されているコマ以外はグレーアウトしています。

2 一段目のコマのアニメーションが終わったら、次の段落のコマへと進み、マンガが一段ずれるようにして止まります。

3 静止した状態から、再びコマの中身がアニメーションしていきます。

サンプルの構造



POINT

1 コマの中のキャラクターのアニメーションです。
この部分では、肩・腕・胴体などのパーツごとに動くアニメーションを作成しています。

2 キャラクターなどのグラフィックシンボルを配置して、コマ全体のアニメーションを作成していきます。ここでは、テキストにトゥイーンアニメーションを使用します。

3 メインステージに各コマを配置して、全体をアニメーションさせます。
マスクの使い方も解説していきます。

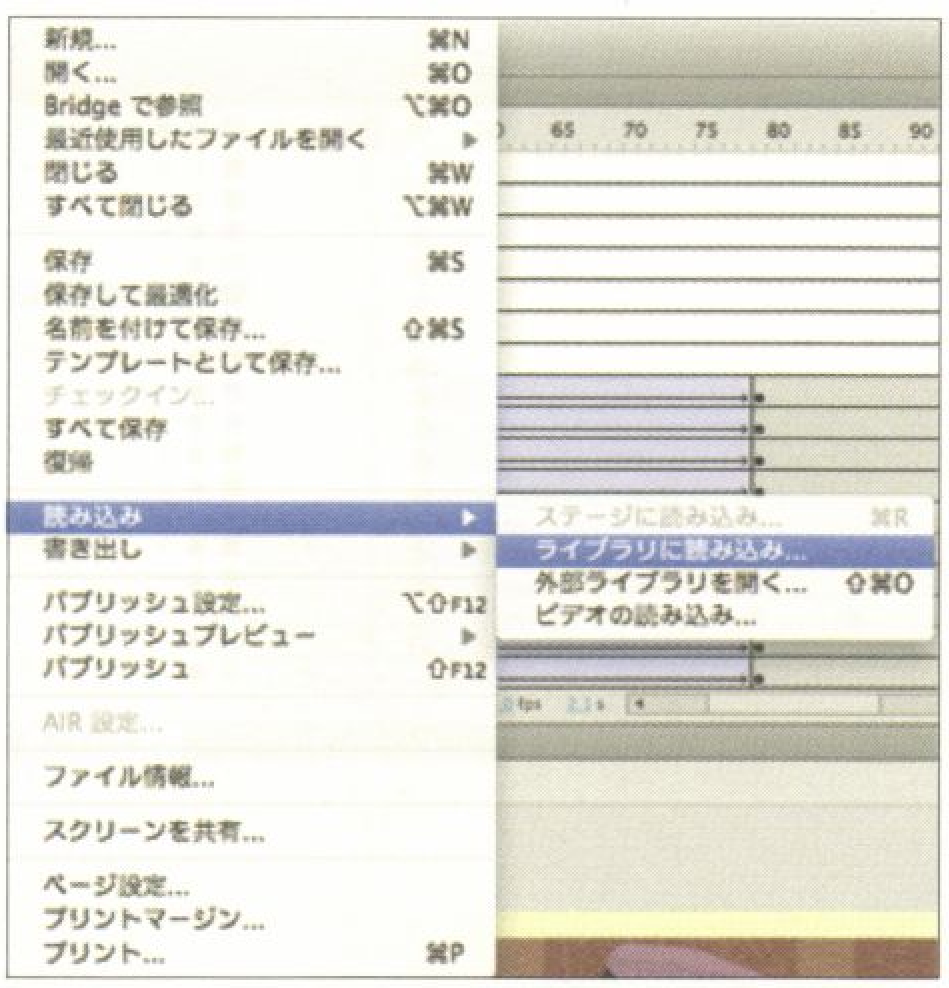
STEP 1　キャラクターのアニメーションを作る

まずは、アニメーションさせる素材を用意して、タイムライン上に配置し、
キャラクターの動きを作ります。

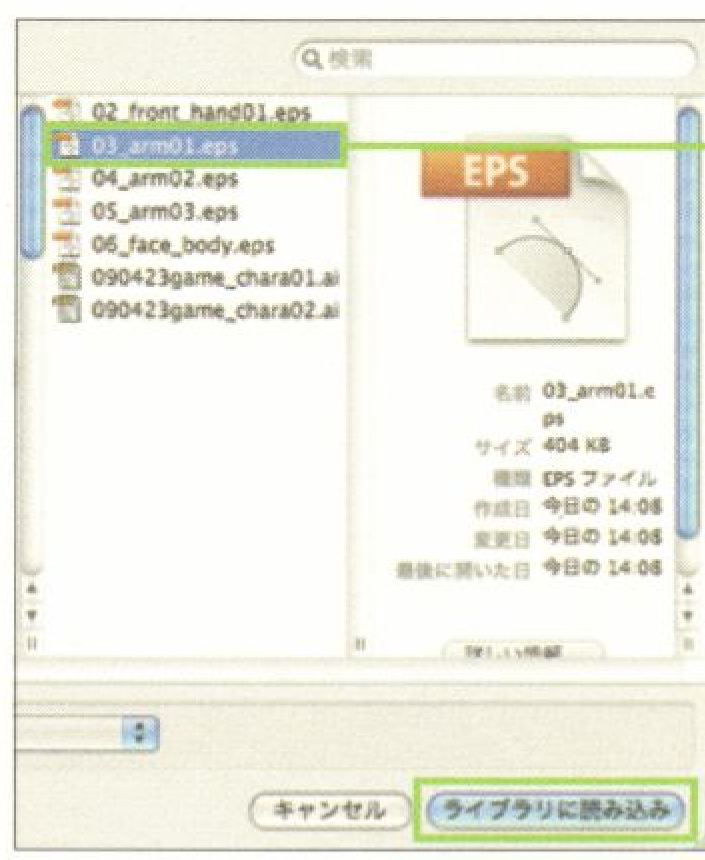


キャラクターの動き

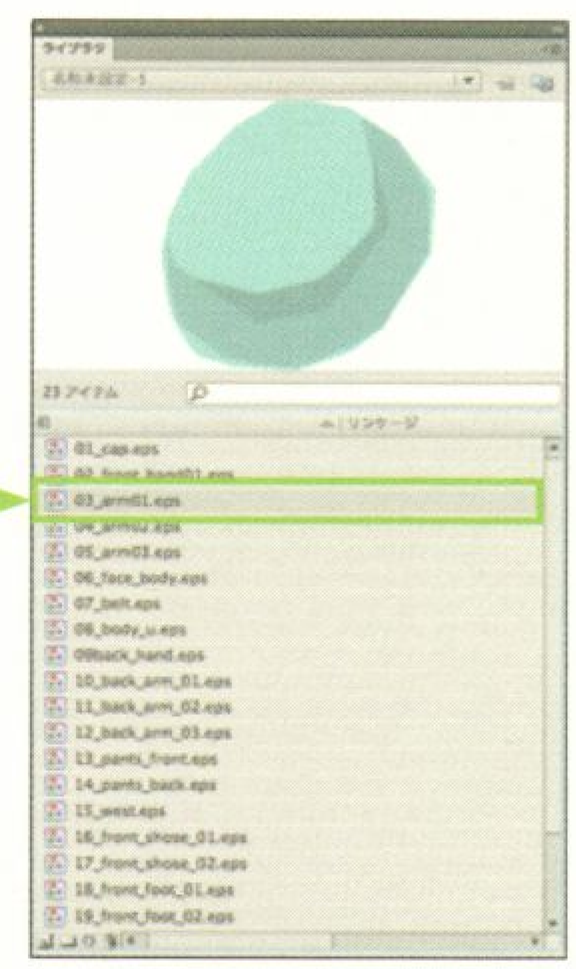
1　まずは要素となるキャラクターの素材をライブラリに読み込みましょう。
[ファイル→読み込み→ライブラリに読み込み] (①) でキャラクターの腕の部分「03_arm01.eps」を選択し、[ライブラリに読み込み]をクリックします(②)。ライブラリに「03_arm01.eps」が読み込まれました(③)。この作業を繰り返して、キャラクターの素材をライブラリに読み込みます。



① [ファイル→ 読み込み→ライブラリに読み込み]

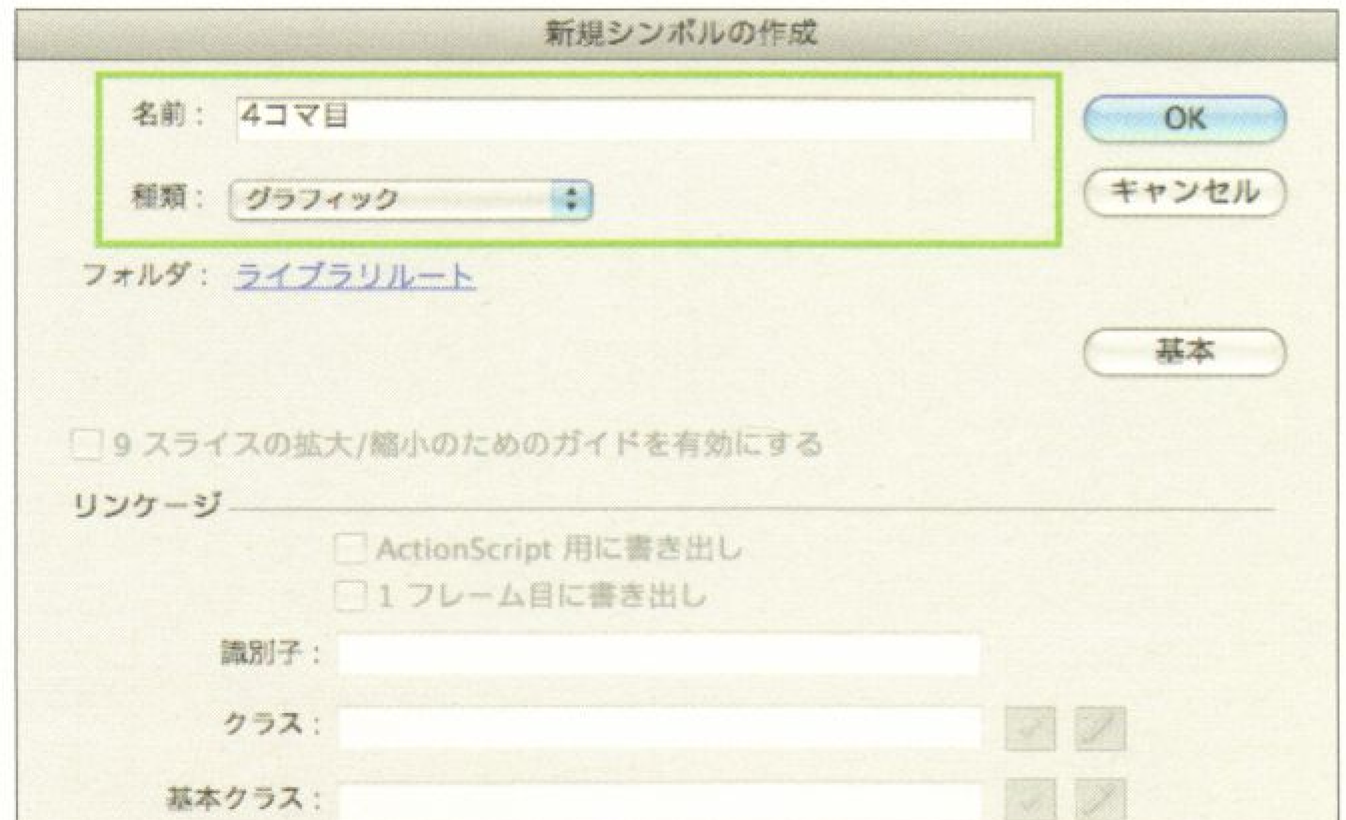


②データ選択→「ライブラリに読み込み」



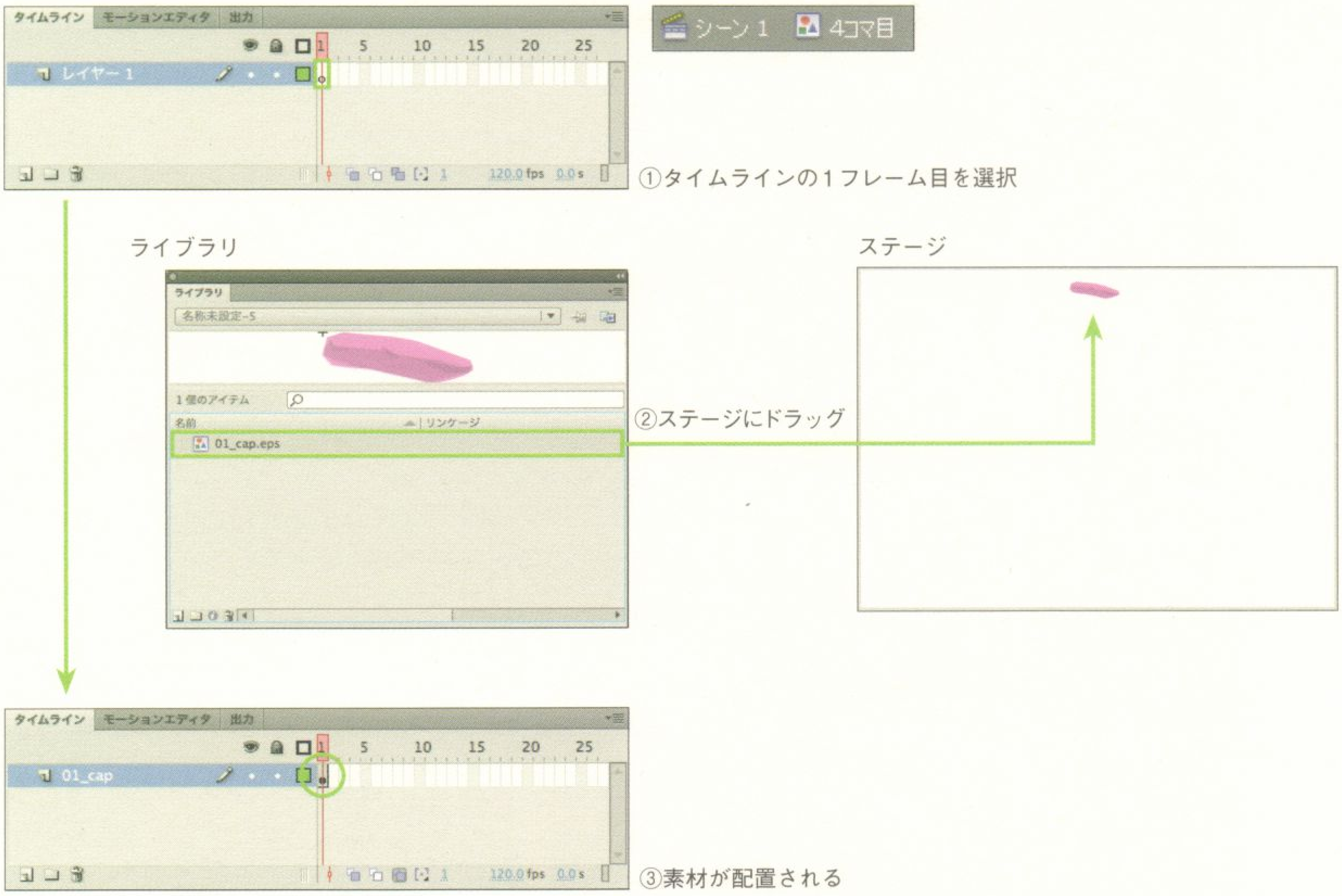
③ライブラリに登録される

2　次に [挿入→新規シンボル] で新規シンボルの作成をします。今回はグラフィックシンボルで作成していきます。[名前] を「4コマ目」として、[種類] は「グラフィック」を選択します。



3

キャラクターの素材を「4コマ目」グラフィックシンボルのタイムラインに配置していきましょう。
「4コマ目」グラフィックシンボルの編集画面にします。タイムラインの1フレーム目を選択し(①)、ライブラリ内の素材をステージ上にドラッグします(②)。
タイムラインの色が黒に変わり、素材が配置されました(③)。



①タイムラインの1フレーム目を選択

ライブラリ

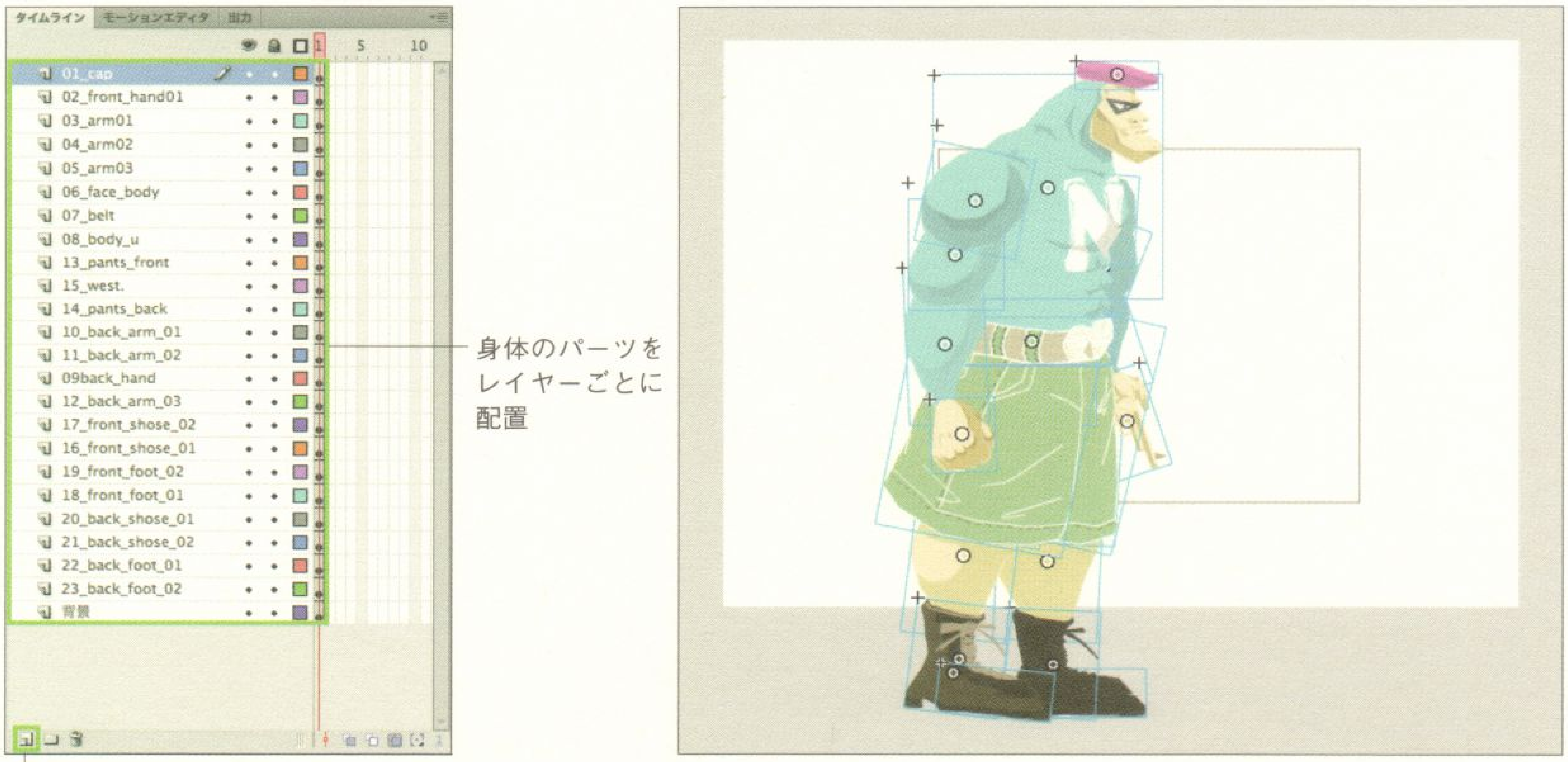
ステージ

②ステージにドラッグ

③素材が配置される

4

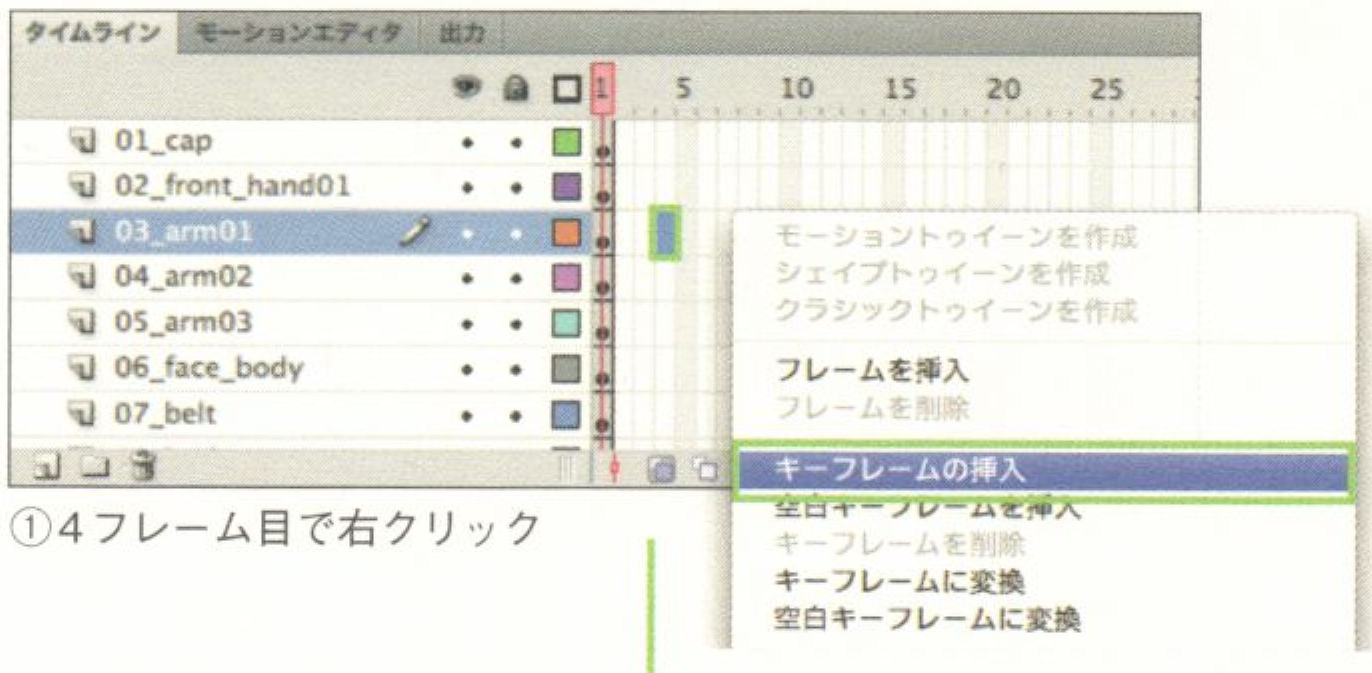
タイムラインの左下のアイコンでレイヤーを複製し、身体の各パーツをレイヤーごとに配置していきます。



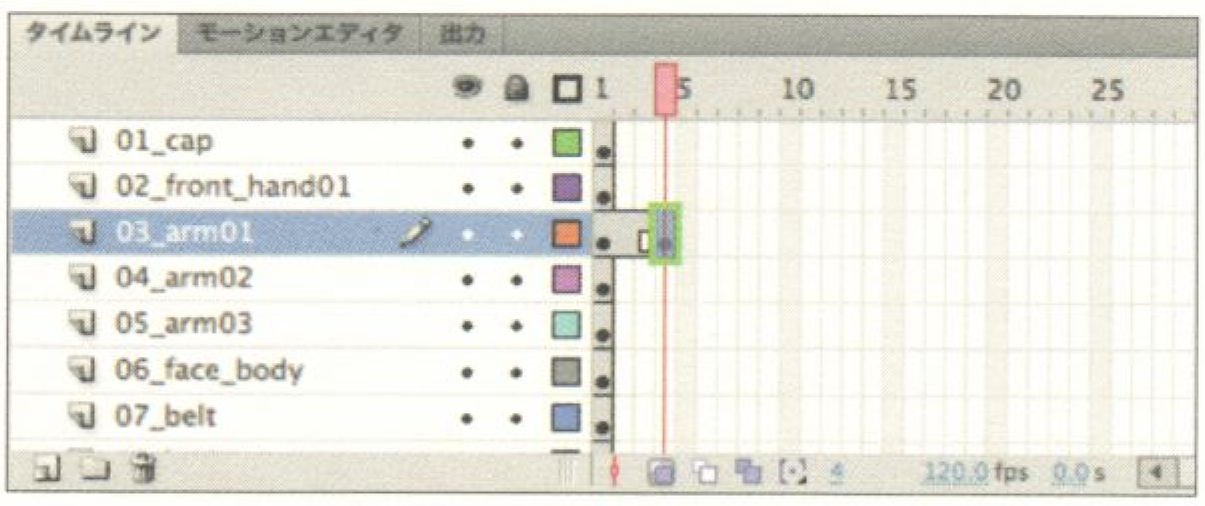
身体のパーツをレイヤーごとに配置

レイヤーを複製

5 アニメーションをさせるフレームを作ります。
4フレーム目のところで右クリックし、[キーフレームの挿入] でキーフレームを作成します(①、②)。
4フレーム目を選択した状態で、ステージ上のオブジェクトを回転させます(③)。数値で回転を変更する場合は、[プロパティ] パネルから「変形」を選択し、「回転」の数値を変更します。

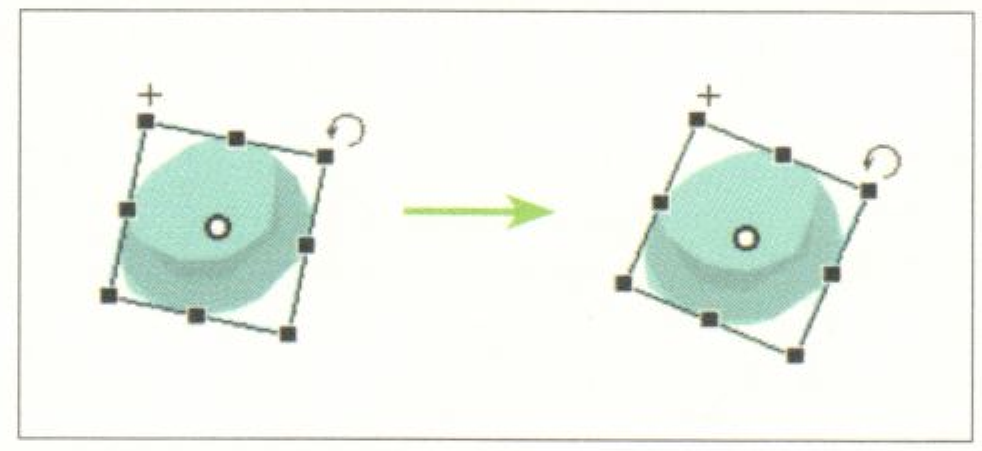


①4フレーム目で右クリック

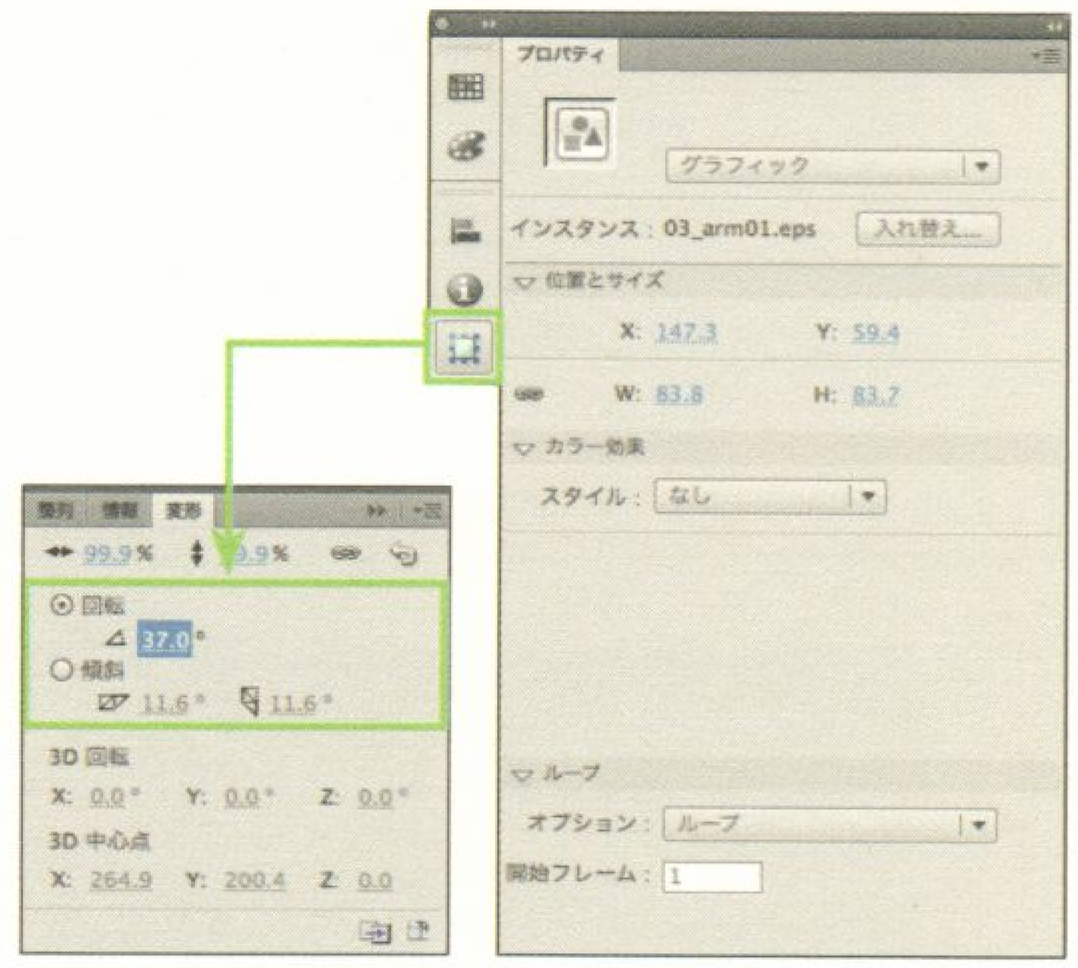


②キーフレームが挿入された

③4フレーム目を選択した状態でオブジェクトを回転



プロパティパネルの数値で回転させる場合

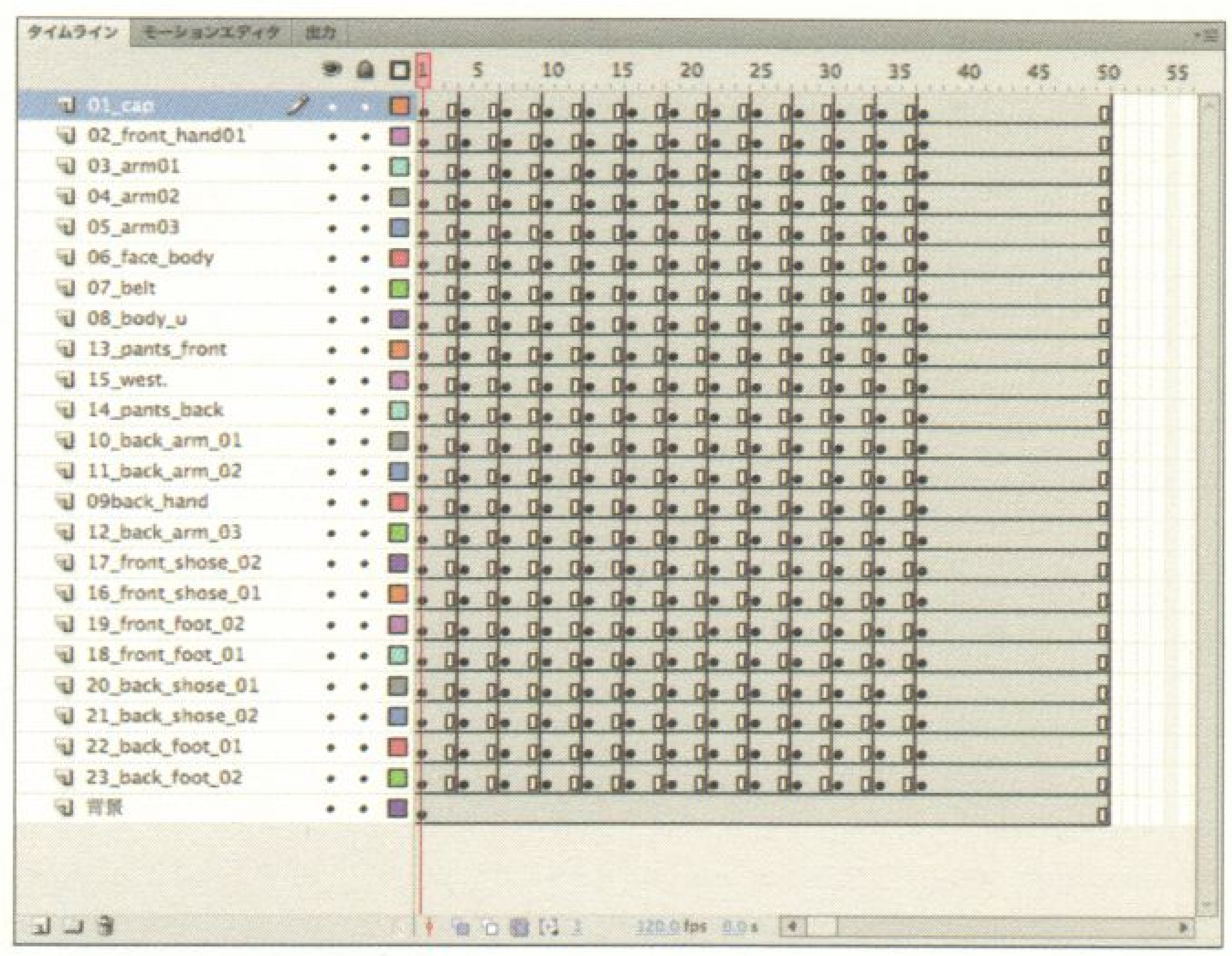


---

6 5 を繰り返してアニメーションをさせたいフレームでキーフレームを作成し、ステージ上の配置を変化させていくとアニメーションが完成します。

■ タイムライン ………………………… P038へ





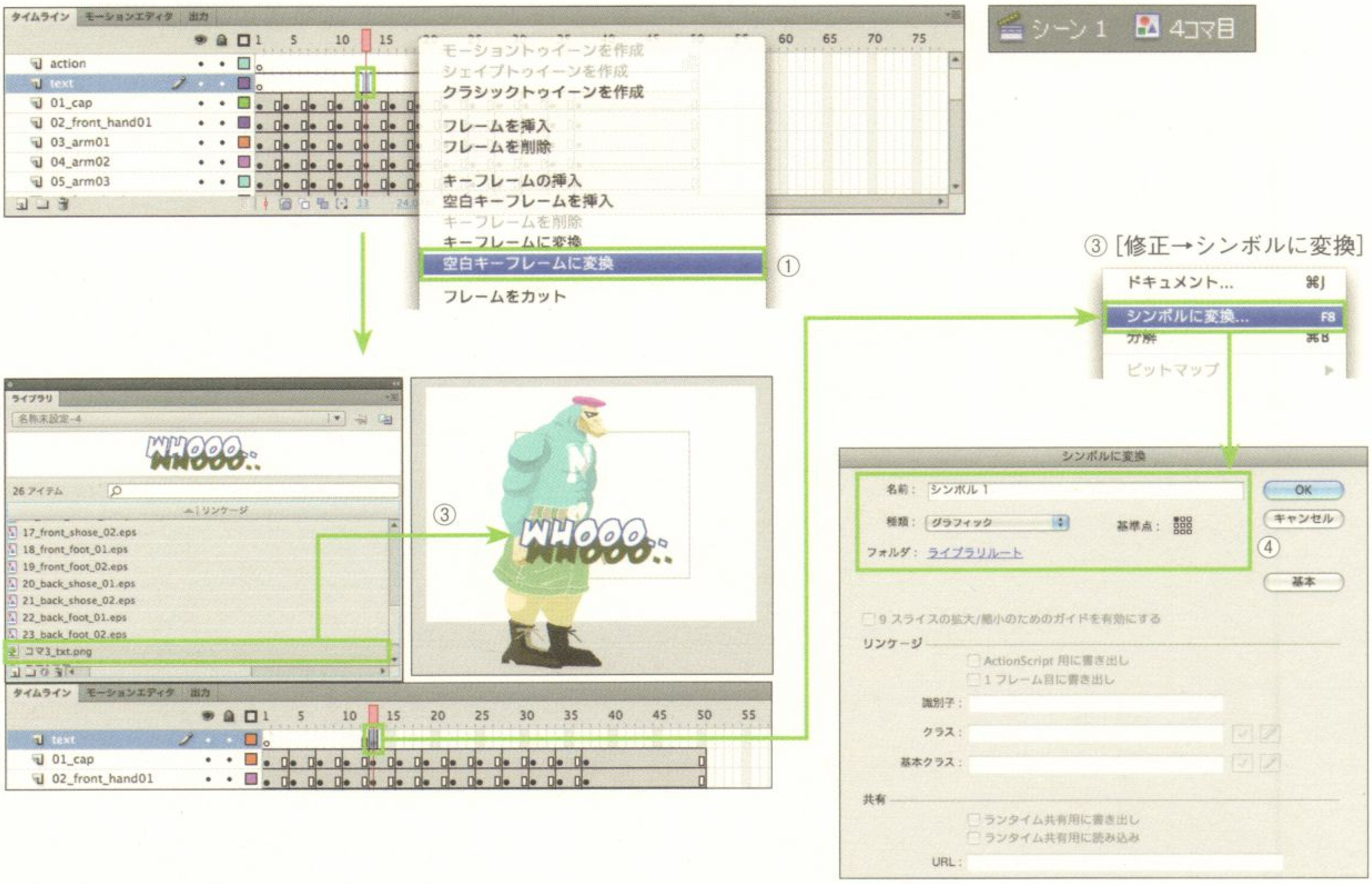
プレビュー



STEP 2 コマ全体のアニメーションを作る

1コマ全体のアニメーションを作成していきましょう。STEP 1の「4コマ目」のグラフィックシンボルに合わせて他のアニメーションも完成させます。今回は、文字の部分をクラシックトゥイーンでアニメーションしていきましょう。

1 まずは、STEP 1と同じ4コマ目のシンボルタイムライン上にレイヤー（「text」）を追加します。次に12フレームから「WHOOO…」の文字が表示されるよう、12フレーム目に「空白キーフレーム」を作成します（①）。12フレーム目を選択した状態で、ライブラリから「コマ3_txt.png」をステージ上にドラッグします（②）。タイムラインは12フレーム目から表示されるようになりました。次に、グラフィックシンボルに変換します。12フレーム目を選択した状態で、メニューの［修正 → シンボルに変換］を選び（③）、「名前：シンボル1」を入力し、「種類：グラフィック」を選択します（④）。



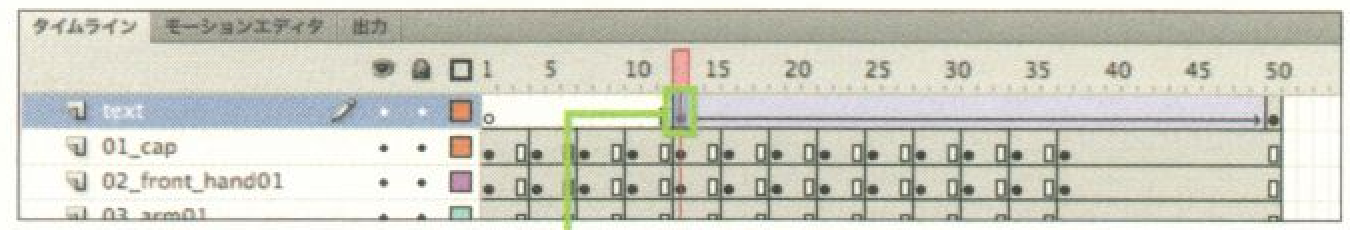
2 「text」レイヤーのタイムラインの50フレーム目を選択し、STEP 1と同じように右クリック（もしくは「挿入」→「タイムライン」）から「キーフレーム」を選択。
同レイヤーの34フーレームを選択し、右クリックで［クラシックトゥイーンを作成］を選択します（①）。タイムラインが紫色に変化し、「→」が表示されるようになりました（②）。



3

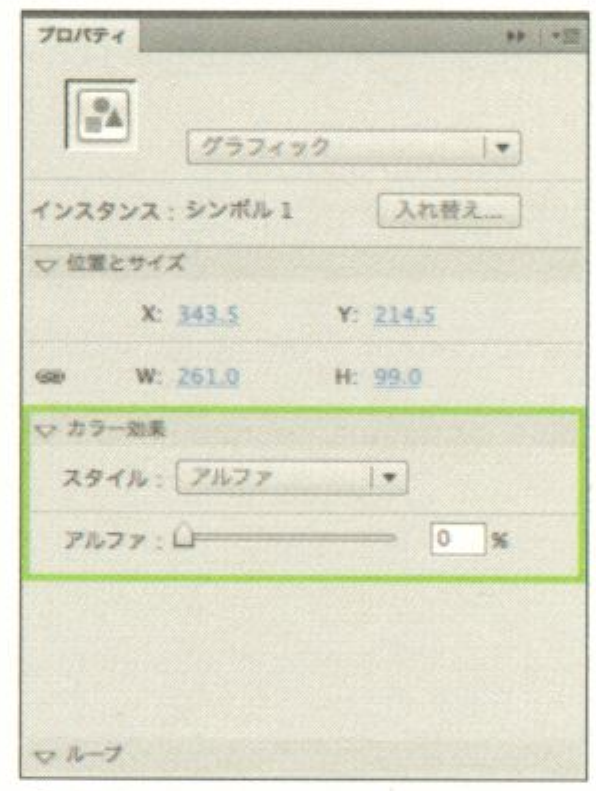
文字を右から左へ動くアニメーションにします。同レイヤーの12フレームを選択します(①)。選択すると、先ほど挿入した文字のオブジェクトのまわりに青い線が現れます。
選択されたら←カーソルキーで左へ10px程度移動します(②)。
さらに、最初のフレームを選択して、[プロパティ]パネルの「カラー効果」の「スタイル：アルファ」で、アルファを0%にします(③)。「アルファ：0%」で完全な透明になりますので、このアニメーションは、文字が徐々に表示されながら右から左へ移動していくものになります。

■アニメーションを作る…………………… P046へ



①12フレームを選択



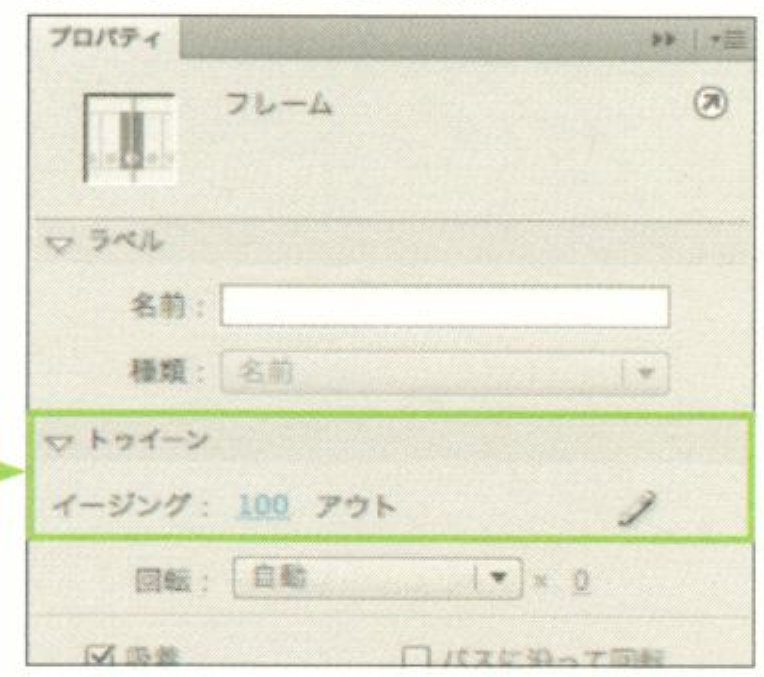


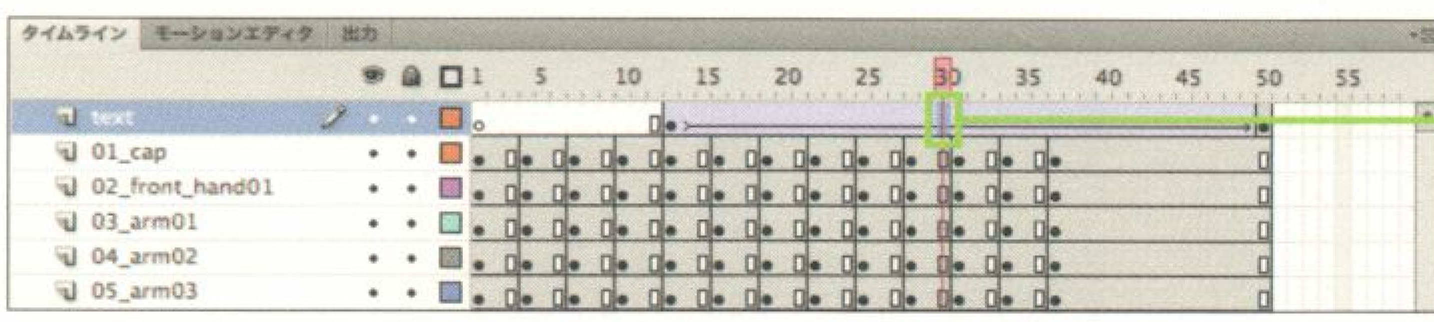
③最初のフレームを選択し、アルファを0%に設定

4

演出を加えるため、アニメーションにイージングを付けてみましょう。作成された文字のトゥイーンの30フレームを選択し(①)、[プロパティ]パネルの「トゥイーン：イージング」を100にします(②)。こうすることで、トゥイーンの速度が減速します。

②イージングを100に設定



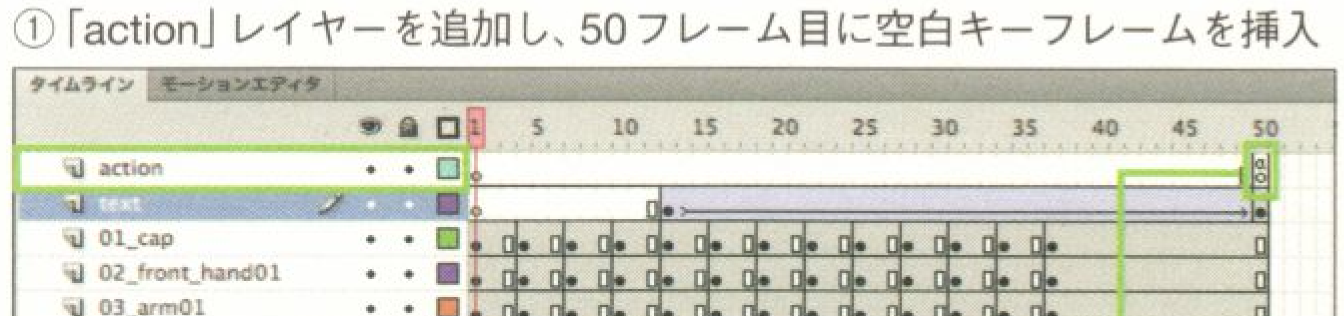


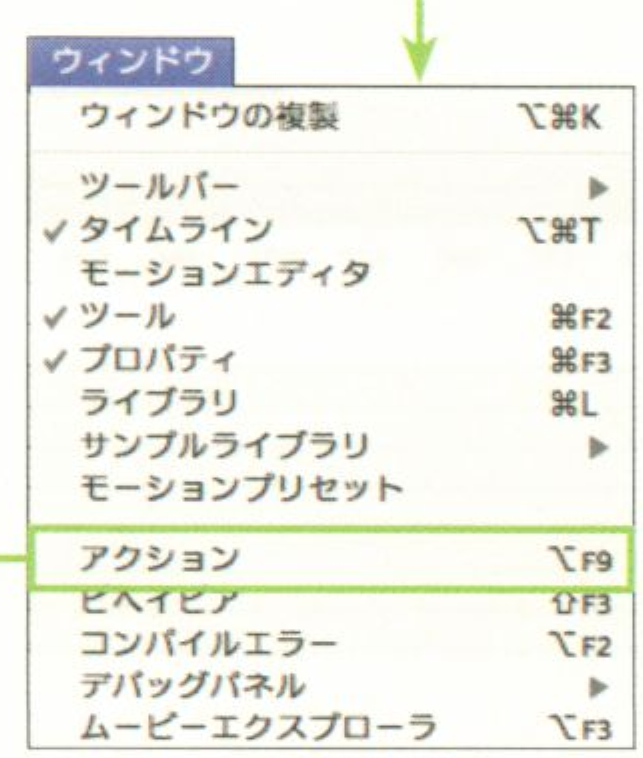
①30フレームを選択

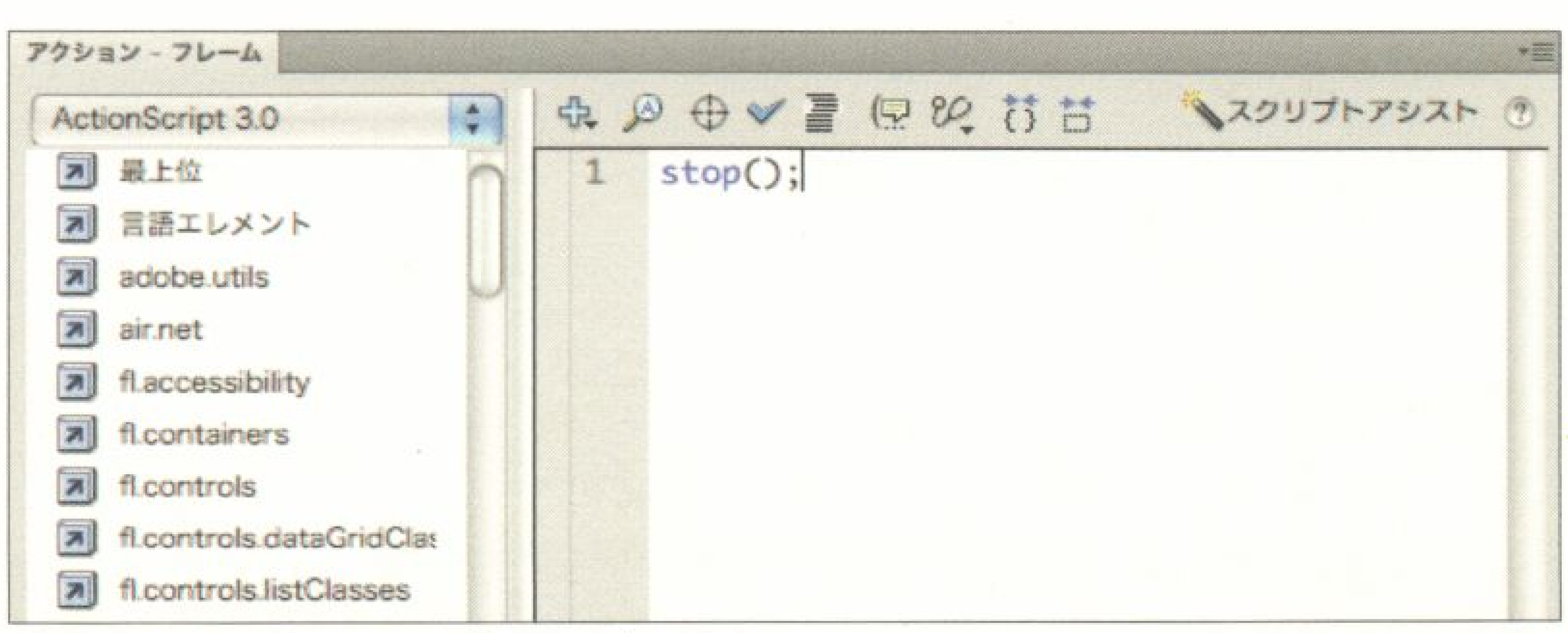
5

アニメーションが終わったらそのまま停止させたいので、「action」レイヤーを追加し、アニメーションの終わる50フレーム目に空白キーフレームを挿入します(①)。50フレーム目を選択して、「ウィンドウ→アクション」で[アクション]パネルを開き(②)、「stop();」と入力します(③)。

①「action」レイヤーを追加し、50フレーム目に空白キーフレームを挿入







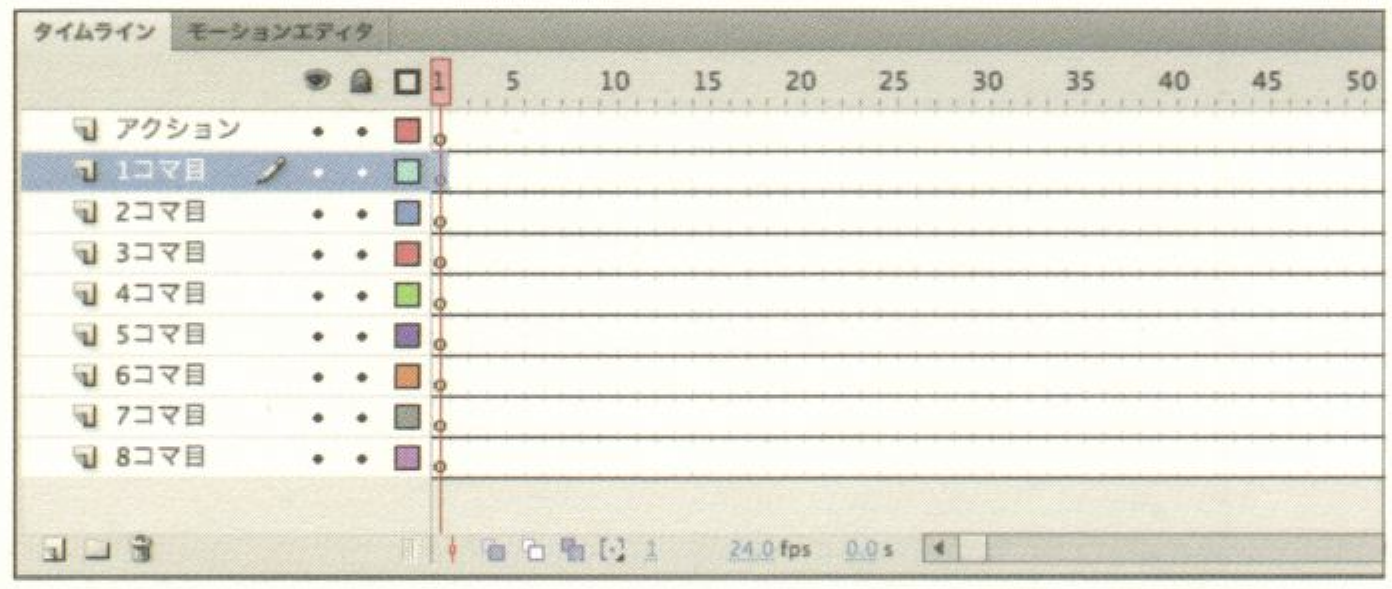
③[アクション]パネルに「stop();」と入力

②50フレーム目を選択し、[アクション]パネルを開く

STEP 3 画面全体のアニメーションを作る

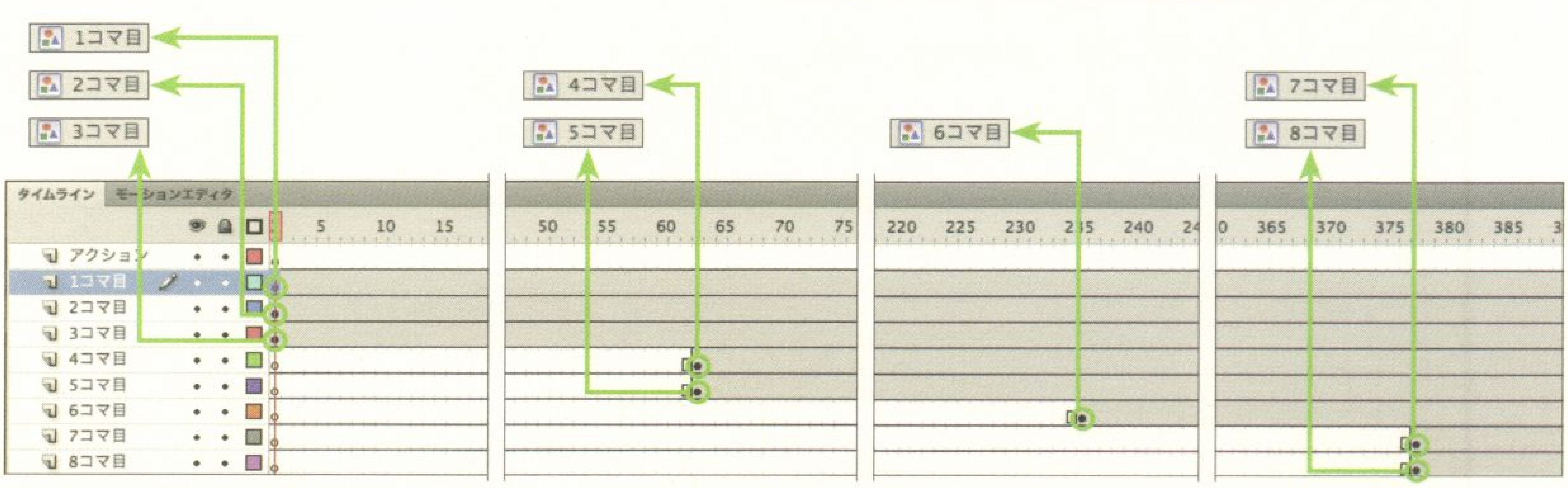
コマの中身になる部分が作成できたら、今度はコマ自体を進めていく部分にアニメーションを付け、入れ子構造のアニメーションを作成します。

1 まずは、1コマ目～8コマ目までのレイヤーを作成していきます。
アニメのコマは全部で8コマですので、メインのタイムラインに、新規レイヤーを8個追加します。



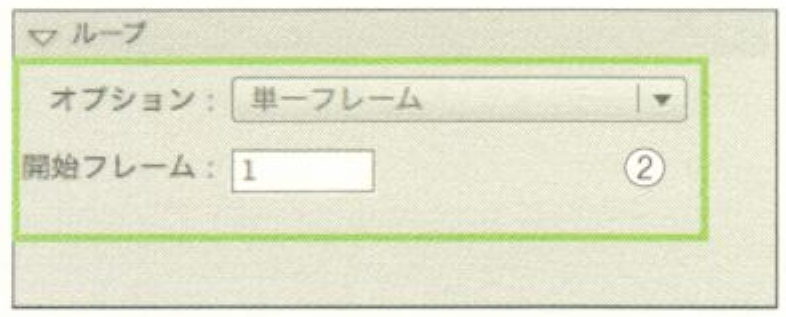
2 ライブラリから、STEP 2で作成したコマ全体のアニメーションの入ったムービークリップを各コマの表示を開始させたいフレームに空白キーフレームを作って配置します。
始めに画面上に現れる1列目のコマ1、2列目のコマ2とコマ3は、1フレーム目に配置。3列目のコマ4とコマ5は63フレーム目に配置。4列目のコマ6は235フレーム目に配置。5列目のコマ7とコマ8は377フレーム目に配置します。





3 コマが縦にスクロールして進んでいくアニメーションを作成していきます。
まず、動かさないコマをグレーアウトさせます。例えばコマ2の場合、1フレーム目を選択、[プロパティ] パネルの「カラー効果」で「スタイル：明度」を選択し、明度を「-60%」にします(①)。
このとき、「ループ」の「オプション：単一フレーム」を選択して開始フレームを「1」にします(②)。「単一フレーム」に設定すると、タイムライン上で1フレーム目だけを静止して表示させます。3コマ目以降についても、表示を開始させたいフレームには同じ設定をしておきます。





例外として、1コマ目は再生の開始フレームになるので、1フレーム目を選択して、[プロパティ] パネルの「ループ」の「オプション:一回再生」を選択し、「開始フレーム」を「2」にします(③)。「一回再生」に設定すると、アニメーションを1回動かすことができます。



「1コマ目」レイヤーは、「オプション：一回再生」「開始フレーム：2」に設定

4 次に縦にスクロールするアニメーションのモーショントゥイーンを設定します。
例えば、1コマ目から2コマ目と3コマ目に進む際、1コマ目が上に一段ずれて、2コマ目と3コマ目がメインで配置されて止まります。同様に2コマ目と3コマ目からに4コマ目と5コマ目に進む際も、2コマ目と3コマ目が上に一段ずれて、4コマ目と5コマ目がメインで配置されて止まります。



5 4コマ目を例にとって、具体的に再生中のコマの設定をしていきましょう。
まず、4コマ目の列の縦にスクロールするアニメーションを設定します。235フレーム目にキーフレームを挿入、258フレーム目で再度キーフレームを挿入します(①)。上の列に従って258フレーム目を上に移動させ、クラシックトゥイーンを設定(②)すると、上にスクロールするアニメーションになります(③)。



①235フレーム目と258フレーム目にキーフレームを挿入

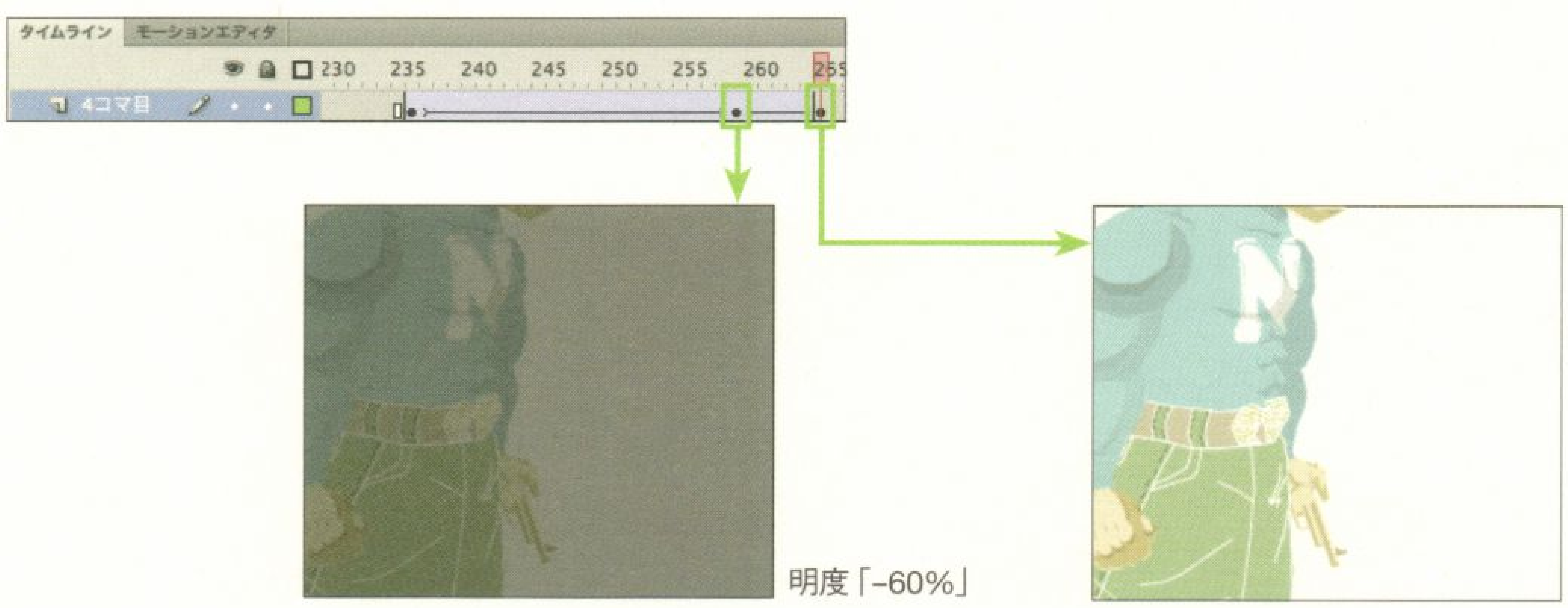


②258フレーム目を上に移動させ、クラシックトゥイーンを設定



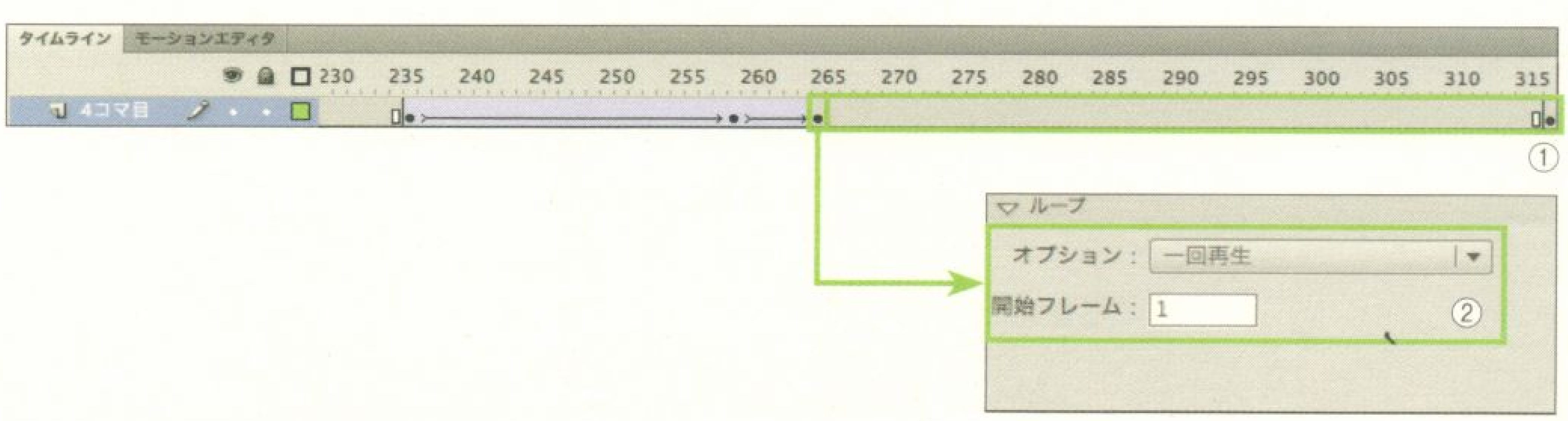
③上に一段ずれて停止するアニメーション(4コマ目)

6 次に、スクロール終了後、前ページの手順3で「-60%」にしていた明度を徐々に「0%」にするアニメーションを設定します。264フレーム目にキーフレームを挿入し、選択した状態で［プロパティ］パネルの「カラー効果」で「スタイル：明度」を選択し、「明度」を「0%」にします。259フレームを選択し、クラシックトゥイーンを設定します。



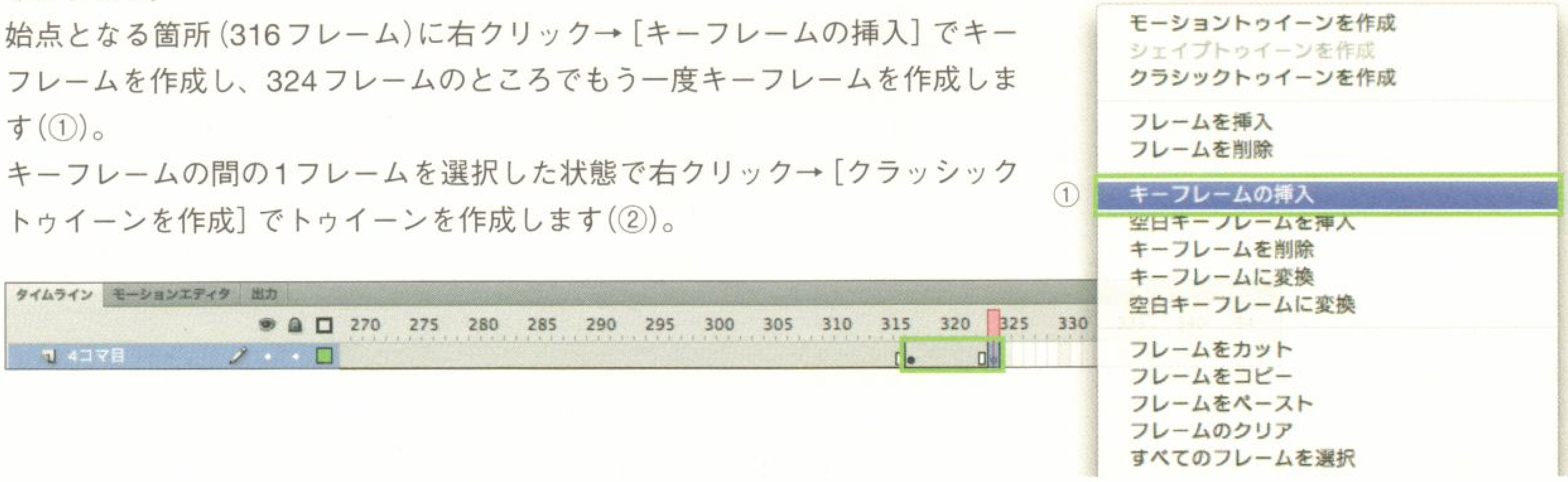
明度「-60%」　明度「0%」

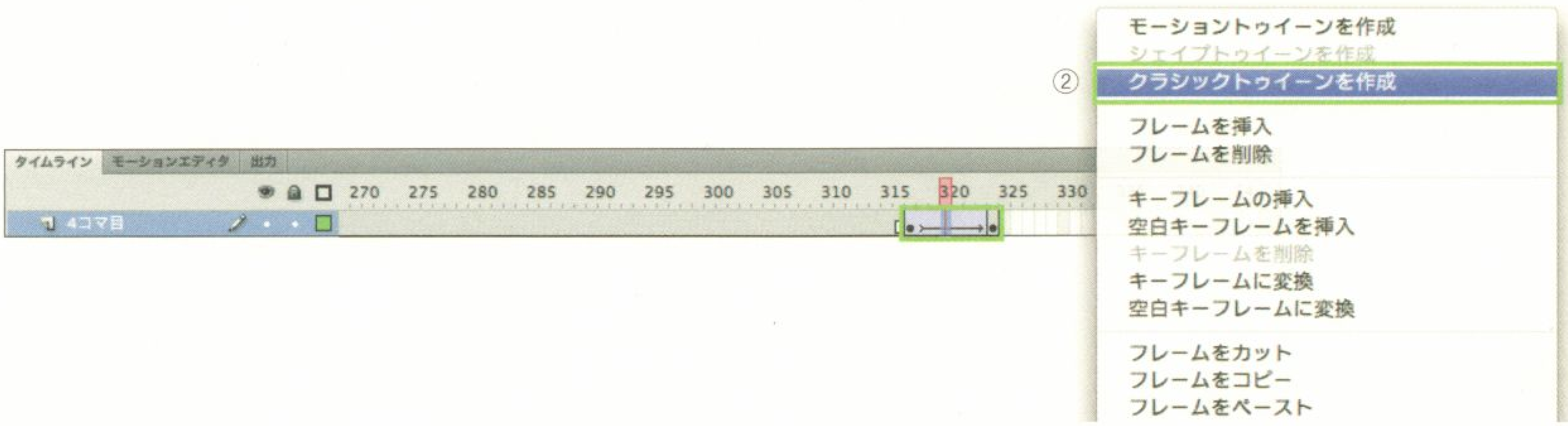
7 次に4コマ目のアニメーションが再生するフレームを追加します。
4コマ目のアニメーションは、フレーム数が50フレームなので、配置したシンボルのフレームを315フレームまで伸ばします。315フレームにキーフレームを挿入します(①)。アニメーションのコマを再生するために、265フレーム目を選択した状態で［プロパティ］パネルの「ループ」で「オプション：一回再生」、「開始フレーム」を「1」にします(②)。



8

次に、アニメーションの終了時にコマがグレーアウトする動きを作成していきましょう。
始点となる箇所(316フレーム)に右クリック→[キーフレームの挿入]でキーフレームを作成し、324フレームのところでもう一度キーフレームを作成します(①)。
キーフレームの間の1フレームを選択した状態で右クリック→[クラッシックトゥイーンを作成]でトゥイーンを作成します(②)。



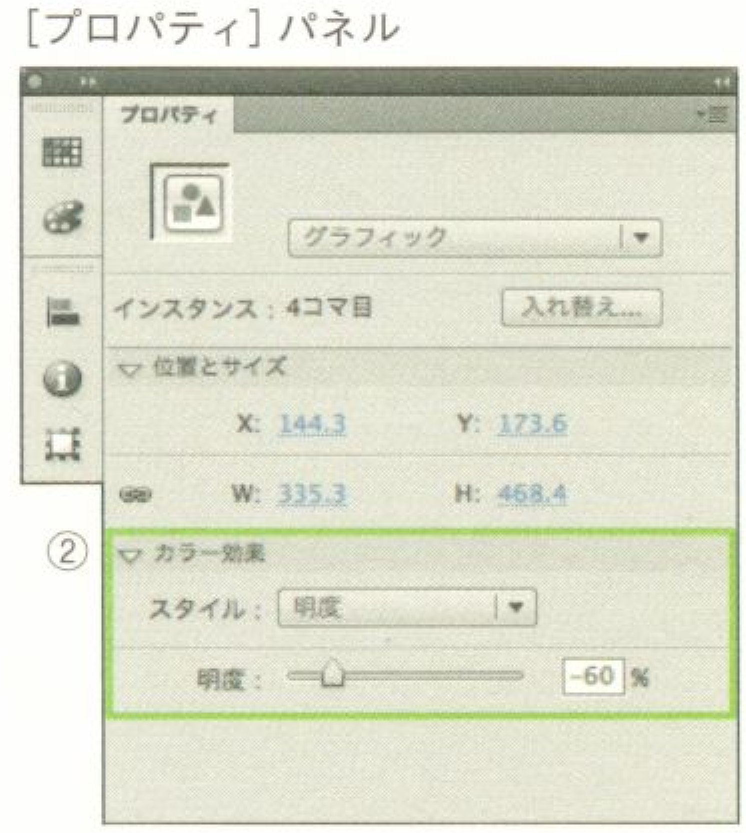


9

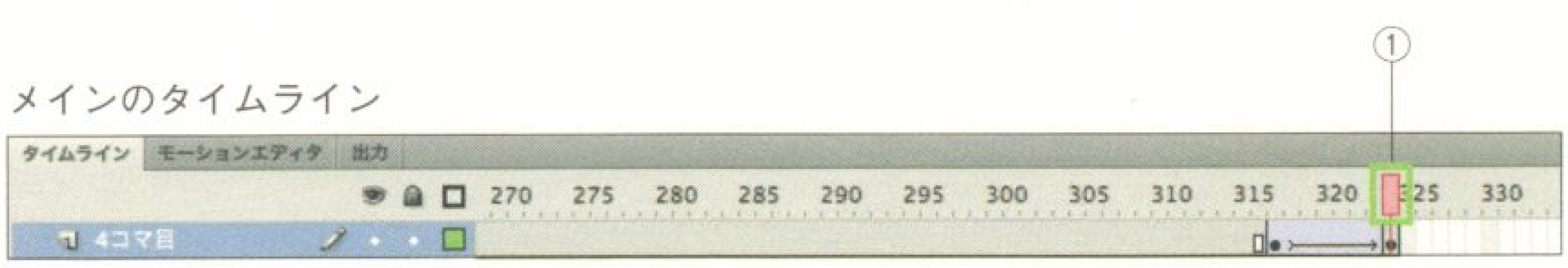
タイムライン上の最後のキーフレームを選択した状態で、ステージ上のグラフィックシンボル「4コマ目」を選択します(①)。
[プロパティ]パネルの[カラー効果]の「スタイル：明度」「明度：-60%」に設定する(②)と、グレーアウトするアニメーションが完成します(③)。

1コマ目～8コマ目の設定も同様に設定してアニメーションを完成させます。

[プロパティ]パネル



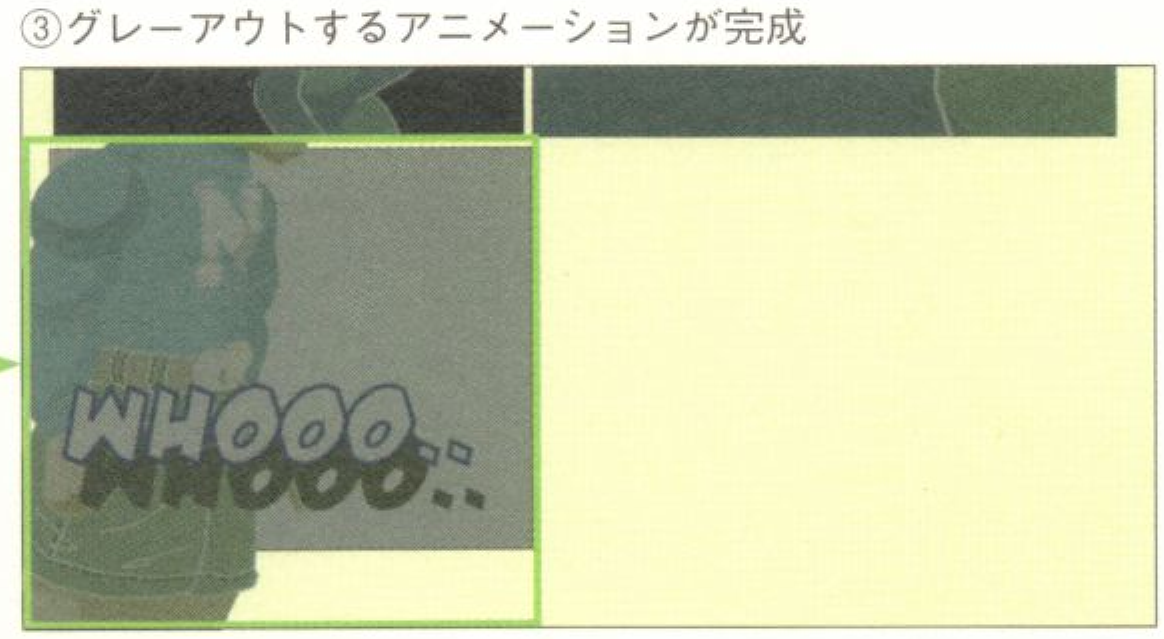
メインのタイムライン



ステージ



③グレーアウトするアニメーションが完成

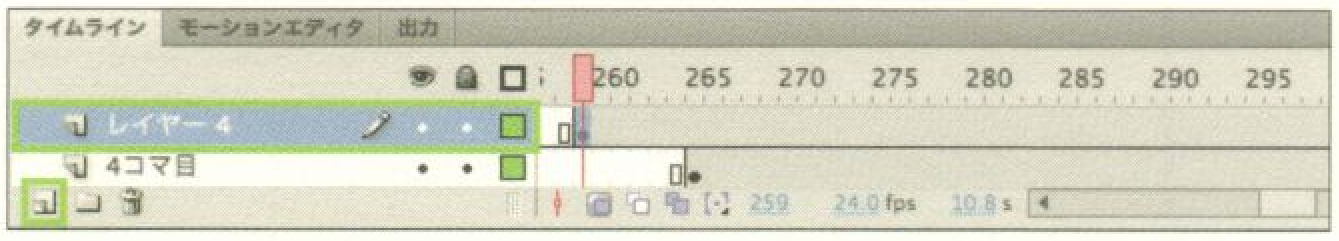


PRACTICE 1

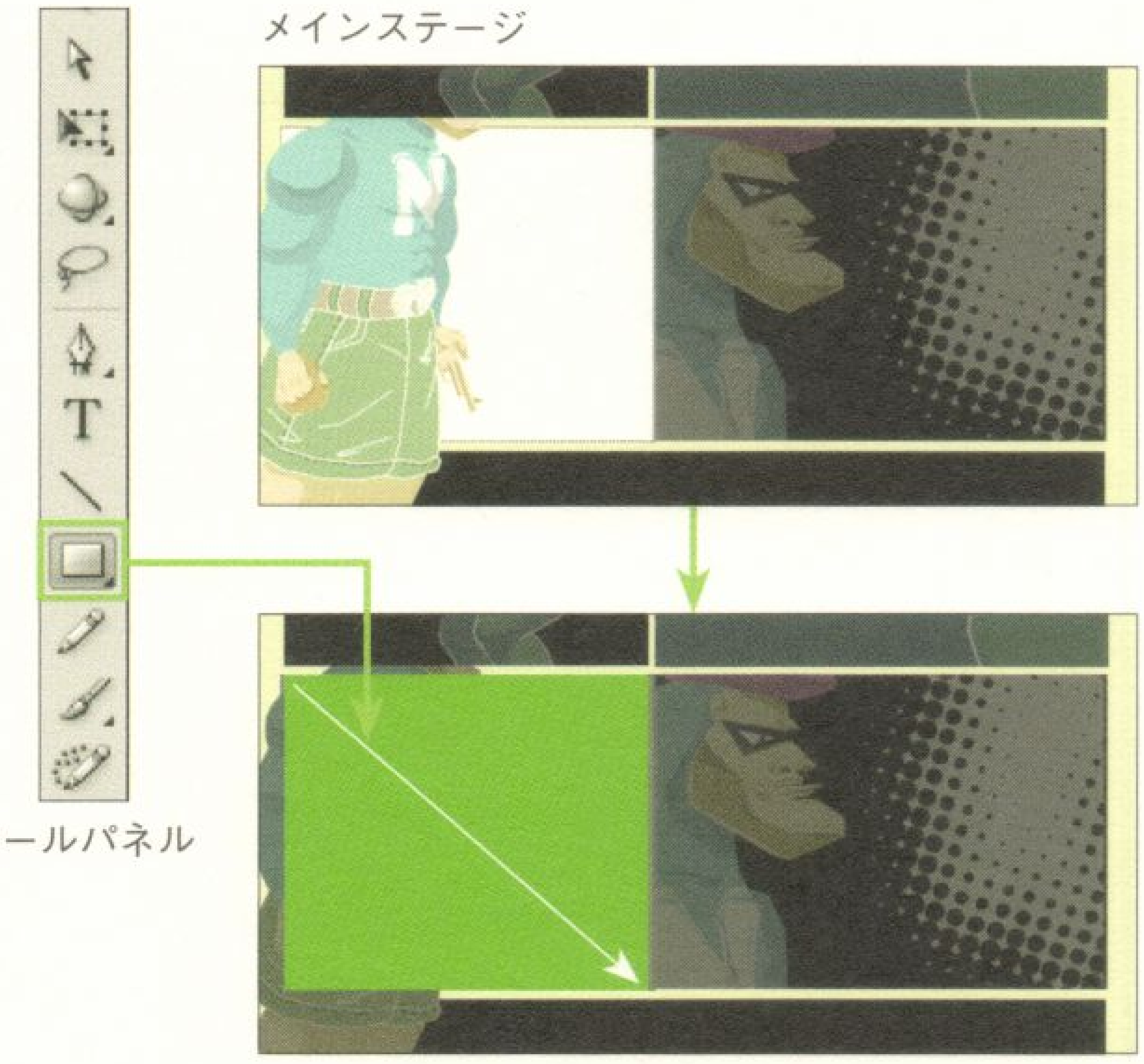
STEP 4 マスクを作成する

最後の仕上げに、コマ全体にマスクをかけていきます。
メインのタイムライン上で作成していきます。

1 タイムライン上のグラフィックシンボル「4コマ目」のレイヤーの上に新規レイヤー［レイヤー4］を作成します。

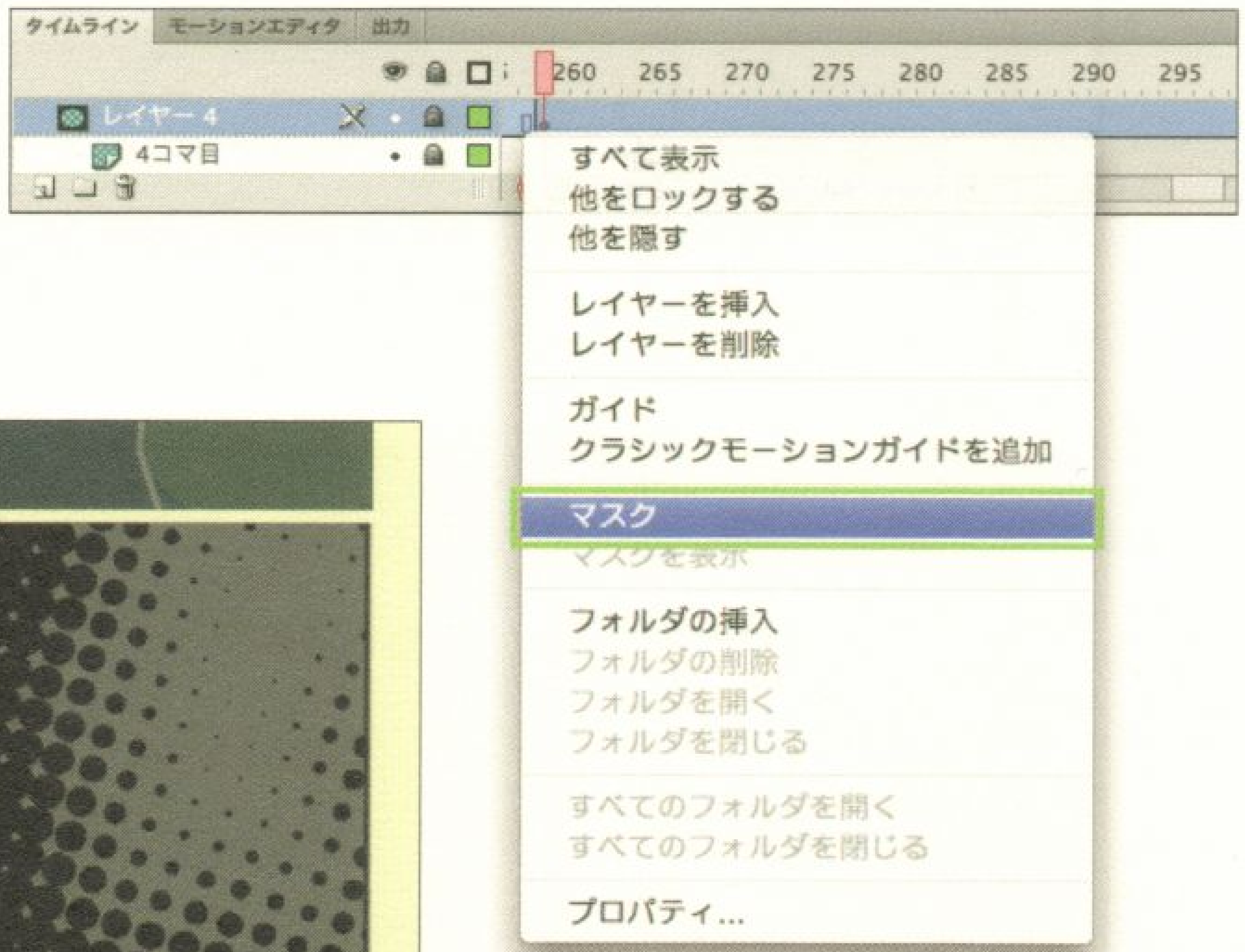


2 新規レイヤー［レイヤー4］の1フレームを選択した状態で、ツールパネルから矩形ツールを選択します。
マスクをかけたい箇所に、矩形を作成します。



3 矩形を作成したレイヤー［レイヤー4］を選択し、右クリック→［マスク］でマスクが適用されます。

■ マスクを使う ……………………………… P051へ



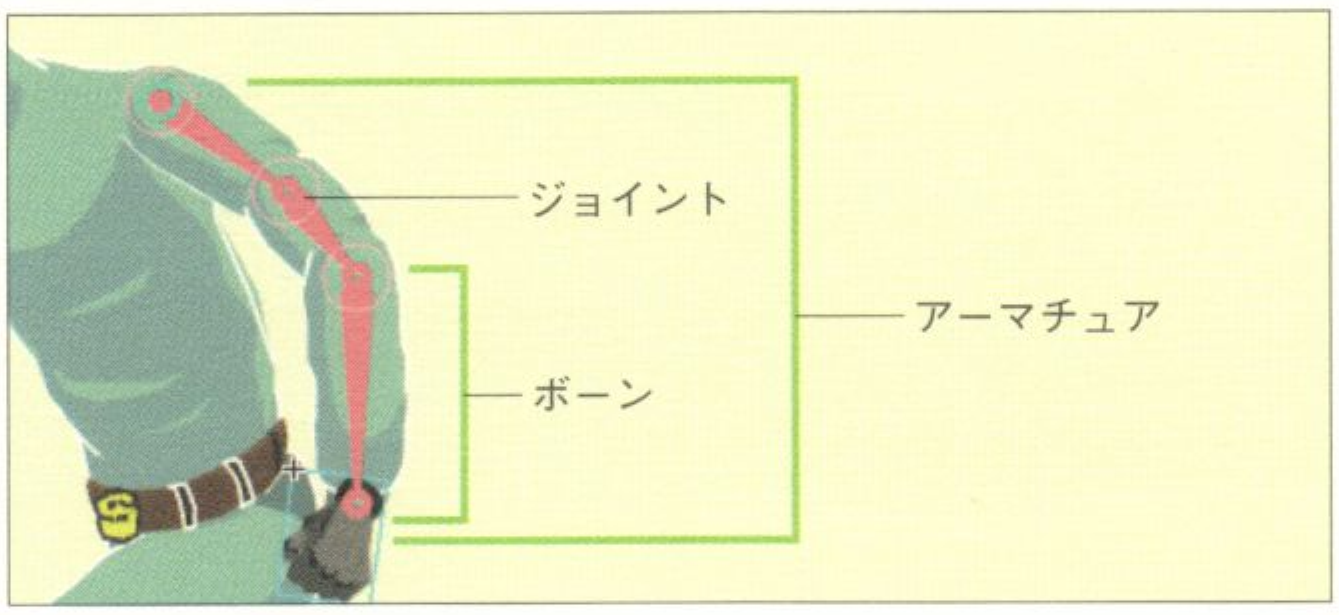


応用

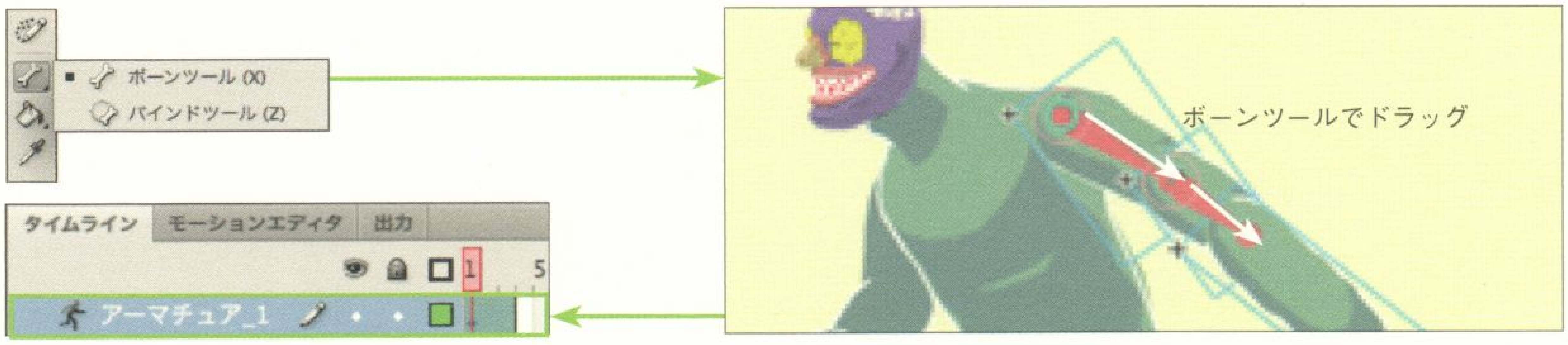
Ex.1 ボーンツールを使用する(CS4～)

Flash CS4から関節(ボーン)の役割をする図形(シェイプやシンボル)やインスタンス(ムービークリップ・ボタン・グラフィック)にボーンツールを使用することができます。今回はインスタンスにボーンを使用する方法を紹介します。

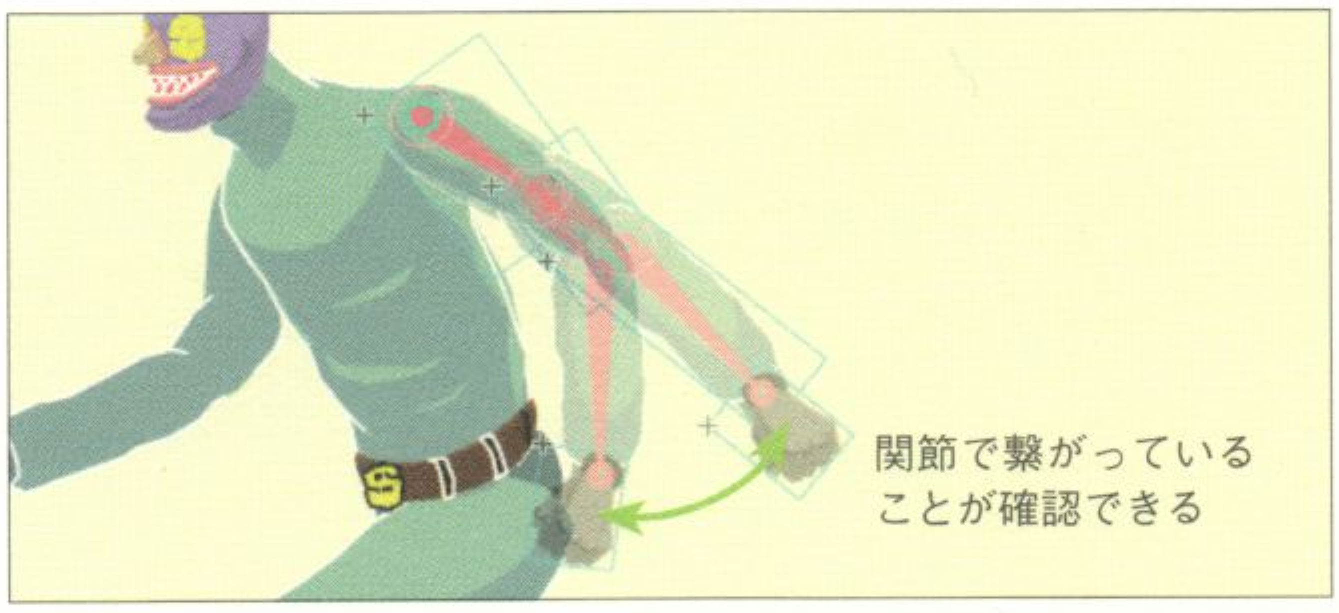
Flashでは、つながったボーン全体を「アーマチュア」、ボーン間を連結する点を「ジョイント」と呼んでいます。



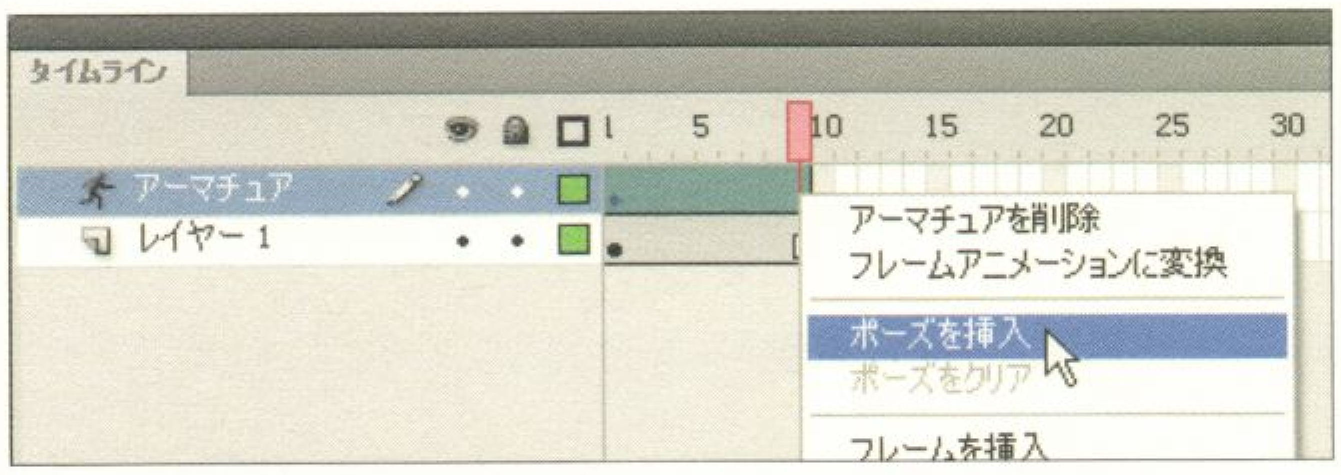
1 ツールパネルからボーンツールを選択し、始点となる関節にカーソルをのせてドラッグします。ボーンが作成されると新しくアーマチュアレイヤーが作成され、ボーンを適用したインスタンスはアーマチュアレイヤーに移動します。



2 選択ツールでボーンを指定したインスタンスを動かすと、このように、関節で繋がっていることが分かります。
作成した各ボーンは、選択ツールで自由に動かすことができ、位置を変更したインスタンスに合わせて、その他のインスタンスも移動、回転して位置を調整します。



3 作成したボーンは、通常のトゥイーンと同様に、タイムライン上でアニメーションを作成することが可能です。タイムライン上で、アーマチュアレイヤーのアニメーションを作成したいフレームを選択し、「ポーズを挿入」します。
ここで作成したタイムラインの「ポーズ」には、それぞれの位置や角度を調整することができ、ポーズ間の動きは自動的にフレーム内のボーンの位置を補完されます。



PRACTICE 2

HTMLページのメニューを作る

表現力：★★☆☆☆　プログラミング力：★★☆☆☆

Webサイトでメニューなどページの一部がアニメーションで動くページを目にすることがよくあると思いますが、それらはFlashとHTMLの組み合わせで作られています。このサンプルでは、そのようなアニメーションを伴うFlashメニューとHTMLページの組み合わせを、架空の旅館のサイトをベースに作っていきます。アニメーションはスクリプトではなくトゥイーンを使うので、あまり難しいプログラミングは必要としませんので、まずはチャレンジしてみてください。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

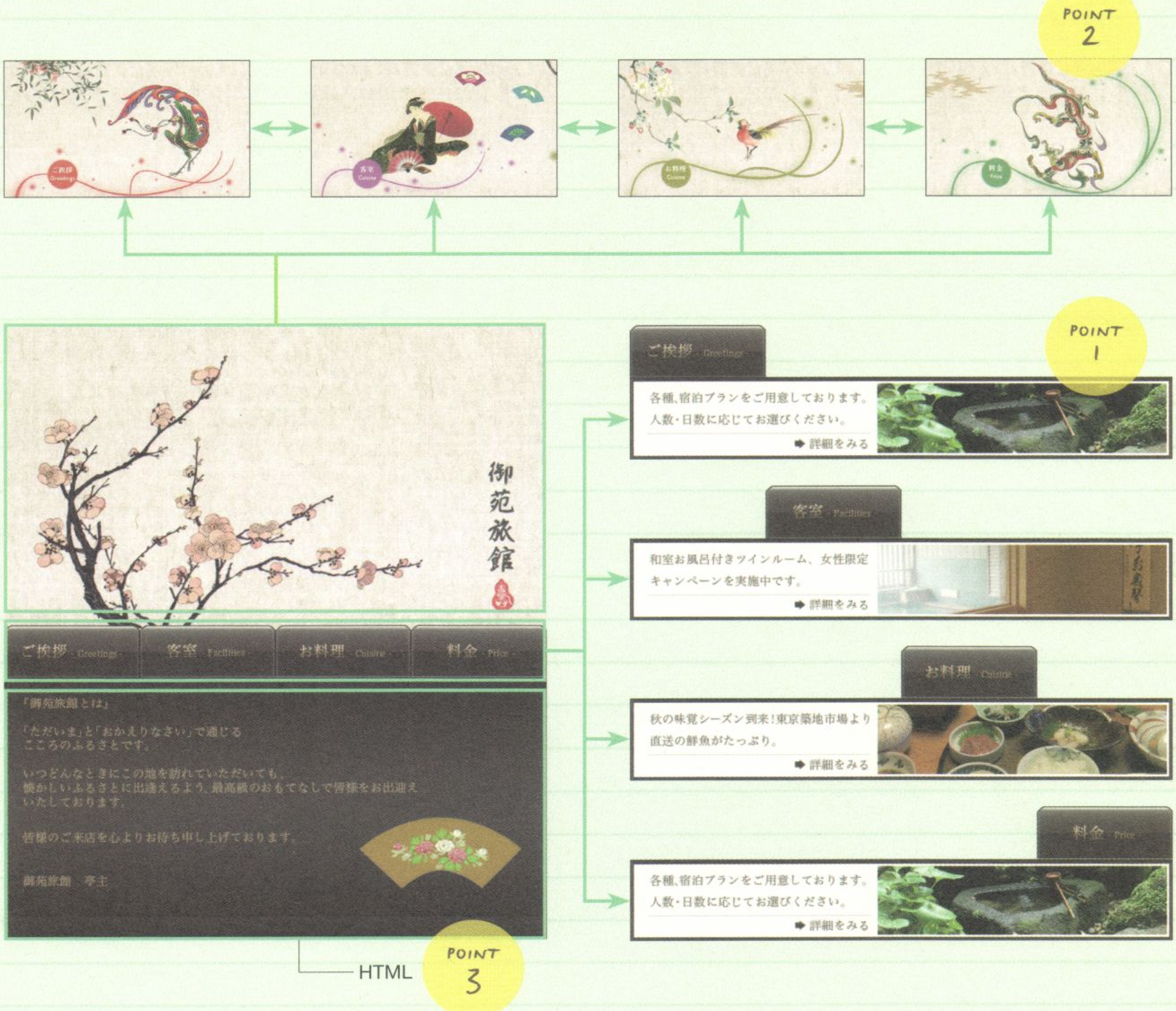
- アニメーションを作る……… P046へ
- イベントの概念……… P077へ
- イベントリスナ……… P090へ
- function……… P086へ
- MovieClipを制御する……… P083へ
- 変数……… P094へ
- if……… P107へ
- navigateToURL( )……… P167へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- 遷移中にボタンが効かないようにする仕組み

サンプルの説明



POINT

1　メニューボタンから上がFlash、下がHTMLページです。メニューボタンの上にマウスカーソルを持っていくと、ボタンがトゥイーンアニメーションでせり上がり、簡単な説明文が表示されます。マウスをボタンから外すと、元の位置に戻ります。

2　ボタンをクリックすると、ボタンの後ろにあるメインイメージがアニメーションでスライドし、そのページのイメージまで移動します。ここでは変数を使って、どのボタンがクリックされているかということを管理し、アニメーションと連携させます。

3　さらに、クリックしたら下のHTMLページが切り替わります。ここではnavigateToURLという仕組みを使って、FlashからHTMLページを遷移させています。

サンプルの構造



POINT

1 全体の構成は、Flashのメニューを置くページと各コンテンツを表示するページを作り、それをフレームを使って表示させている形です。

2 Flashのメインタイムラインは1フレームのみです。メインイメージのムービークリップが置かれているレイヤーとボタンのムービークリップが置かれているレイヤー、そしてActionScriptを書くレイヤーの計3つのレイヤーがあります。

3 メインイメージのムービークリップは、ボタンをクリックした際のスクロールのアニメーションを実現させるために、1枚の横に長いムービークリップを入れ子で配置したものをトゥイーンで動かします。ActionScriptでトゥイーンのスタート／ストップを管理するため、スクロールのパターンの数だけトゥイーンを作成しています。

4 ボタンはマウスオーバー／アウト時のアニメーションをそれぞれトゥイーンで作成しています。

STEP 1 画面に要素を配置する

まずは制作の下準備です。メインタイムラインの構成を決め、ムービークリップを画面上に配置しましょう。大きく分けて、メインイメージのムービークリップと、ボタンのムービークリップを配置します。

1 メインタイムラインには「スクリプト」、「ボタン」、「メインイメージ」の3つのレイヤーを作ります。



2 ステージ上にメインイメージとボタンのムービークリップを配置します。メインイメージはインスタンス名を「mainImage_mc」とし、その中にはトゥイーンで動かすために横1枚の画像のムービークリップ「mainImage_image」を入れ子にしておきます。スクロールする際に便利になりますので、ムービークリップを作成する際、基準点は左上にしておきましょう。メインタイムラインの「メインイメージ」レイヤーで、mainImage_mcを配置しておきます。

■ シンボル ………………………………… P034へ



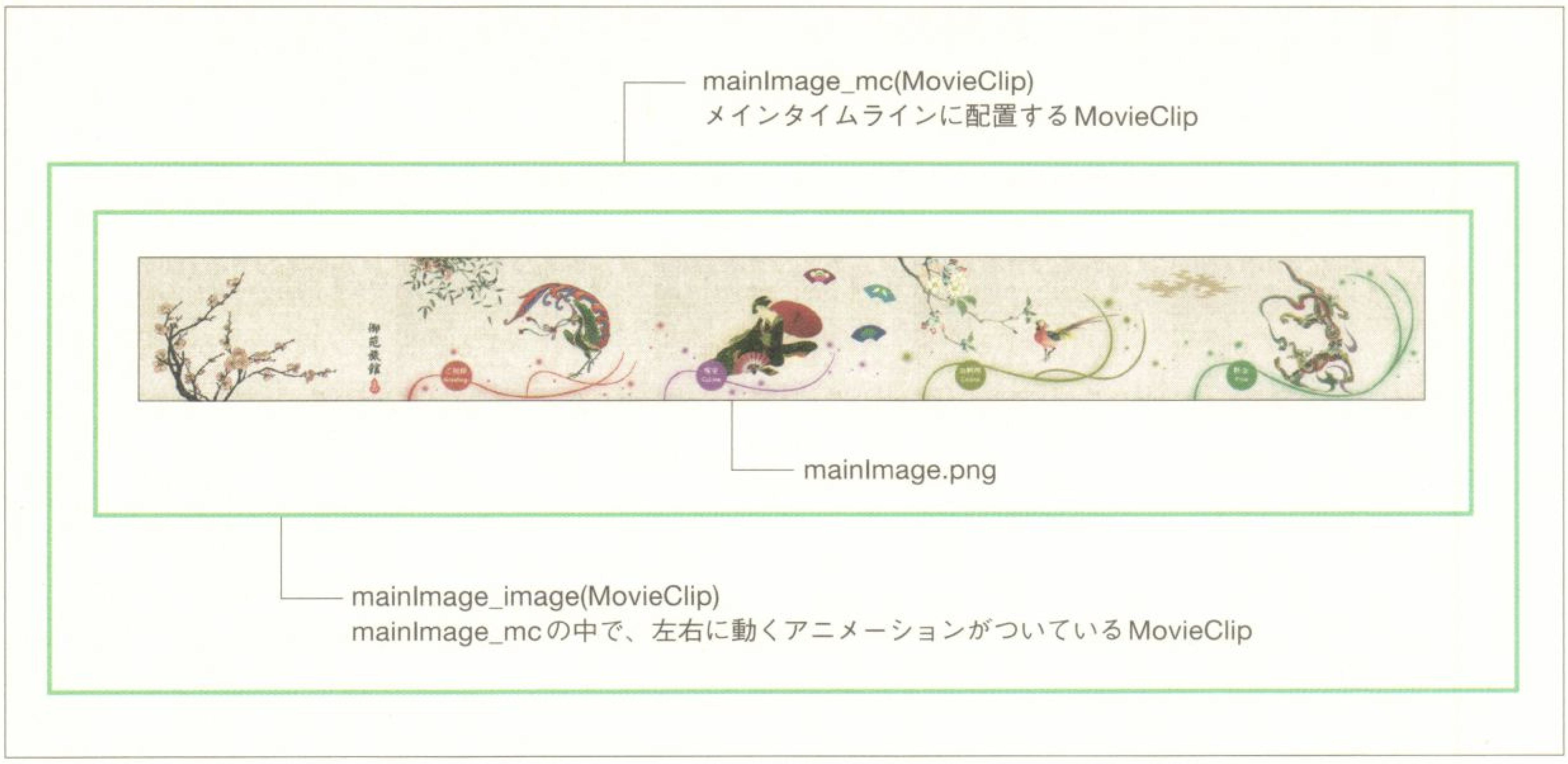
ボタンは1つひとつ別々のムービークリップとして作成します。インスタンス名を「btnA（〜D）_mc」とし、メインイメージ同様、トゥイーンで動かすためのムービークリップ「btnA（〜D）_image」を入れ子にしておきます。
メインタイムラインの「ボタン」レイヤーで、ステージ上にbtn_A.mc〜btnD_mcを重ねて配置しておきます。重なり順はあとでActionScriptでコントロールするので、ここでは気にしなくても大丈夫です。



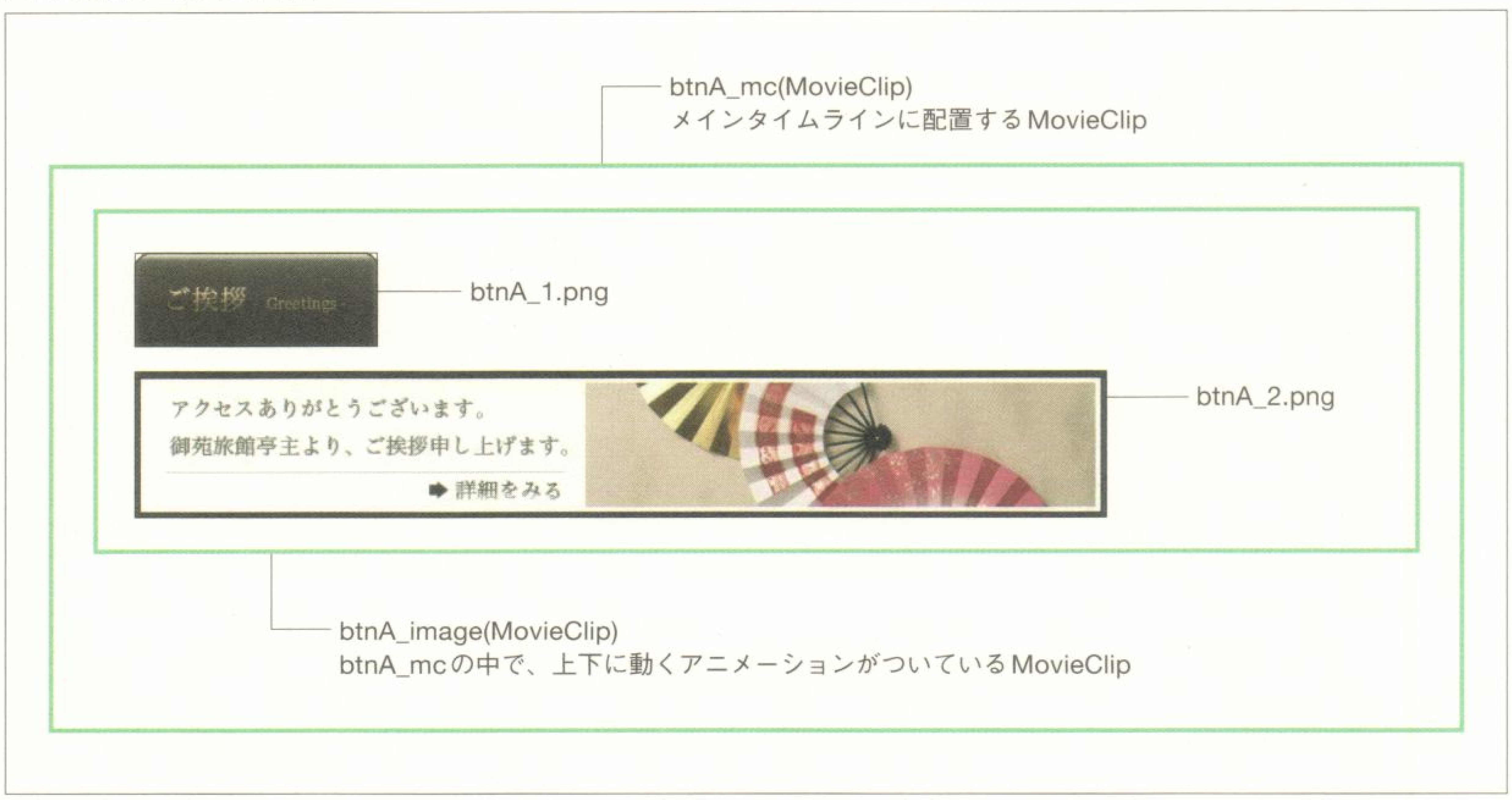
btnA_mc　btnC_mc
btnB_mc　btnD_mc

3 mainImage_mcとbtnA_mcの入れ子構造を図にすると、以下のようになります。ムービークリップの入れ子はFlashではよく使うので、構造をしっかりイメージしながら制作を進めることが重要です。

mainImage_mcの入れ子構造



btnA_mcの入れ子構造

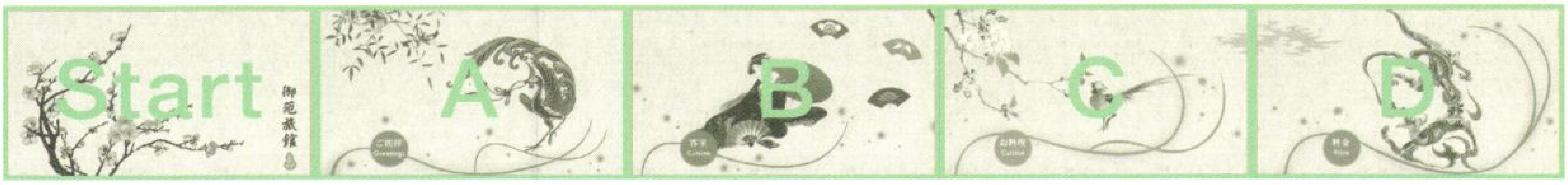


STEP 2 メインイメージのアニメーションを作る

最初に、ボタンをクリックしたときにメインイメージがスクロールするアニメーションを「mainImage_mc」の中で作ります。ポイントは、スクロールするパターンの数だけトゥイーンを作り、プログラムで制御できるようにフラグを付けておくことと、現在表示されている場所の情報を変数で持たせることです。

1 まずはメインイメージのすべての表示パターンを考えます。サンプルでは、「トップページ」、「ご挨拶」、「客室」、「お料理」、「料金」の5つの画面と、トップページ以外の4つのメニューボタンがあり、それぞれの画面から他の画面にスクロールする組み合わせは5×4＝20通り、またページを読み込んだときに最初に表示されるトップページの画像が静止している状態を含めると、計21通りの表示パターンがあることが分かります。このパターンすべてをmainImage_mcの中に作っていきます。また、あとでActionScriptでスクロールを制御する際に分かりやすくするために、スクロールさせるムービークリップのそれぞれの画面に対応する部分を「Start」「A」「B」「C」「D」と分けて考え（下図）、レイヤー構造は、右図のようにしておきます。



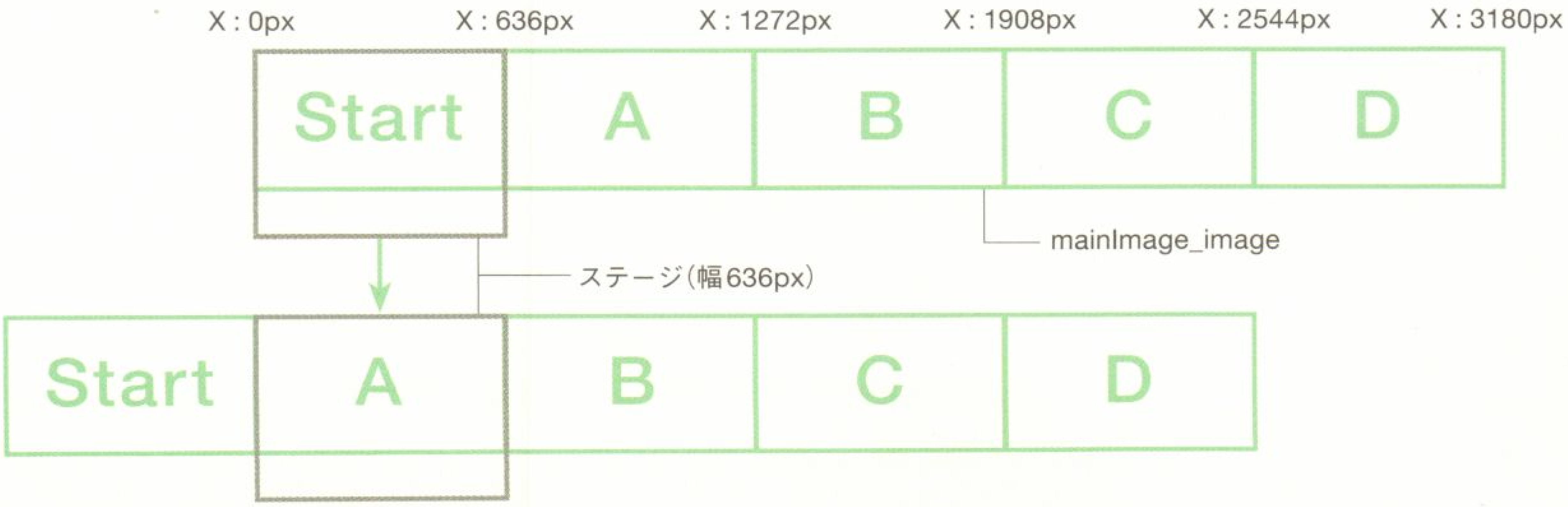


2 「Start」レイヤーにmainImage_imageを配置し、「Start」で静止している状態（1〜10フレーム目）、そして「Start」から「A」、「B」、「C」、「D」と遷移するトゥイーンアニメーション（11フレーム目から、20フレームずつ）を作ります。それぞれのトゥイーンにはアウト側に100のイージングを付けています。
「A」〜「D」のレイヤーに対しては静止状態は作らず、それぞれ他の画面にスクロールするアニメーションを作ります。

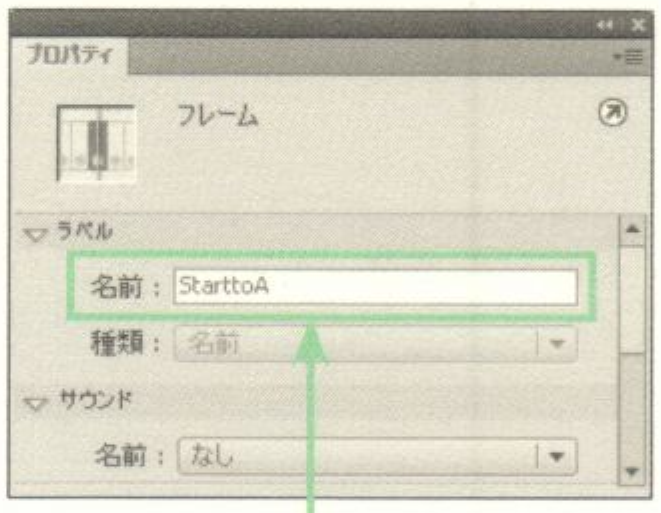
■ アニメーションを作る …………………… P046へ
■ イージング …………………… P041へ

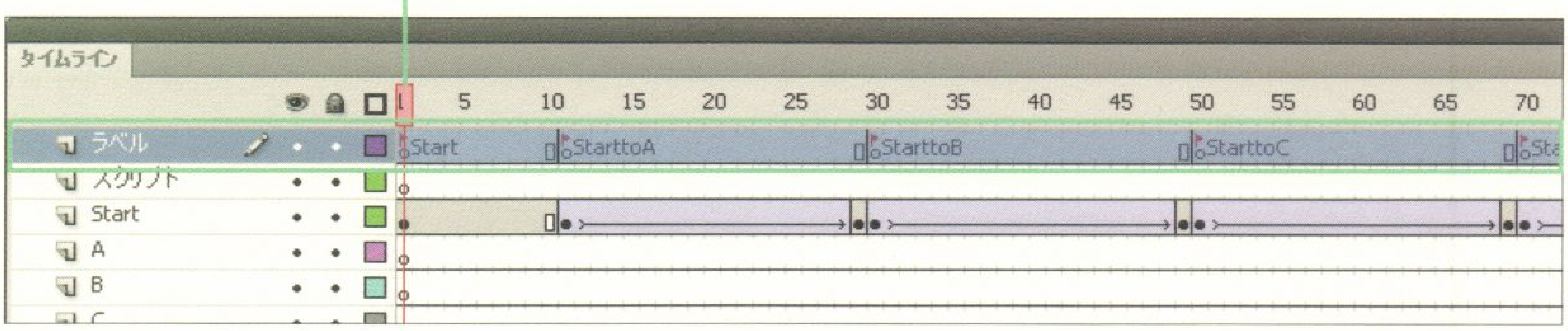


mainImage_imageが「Start」→「A」にスクロールする動き

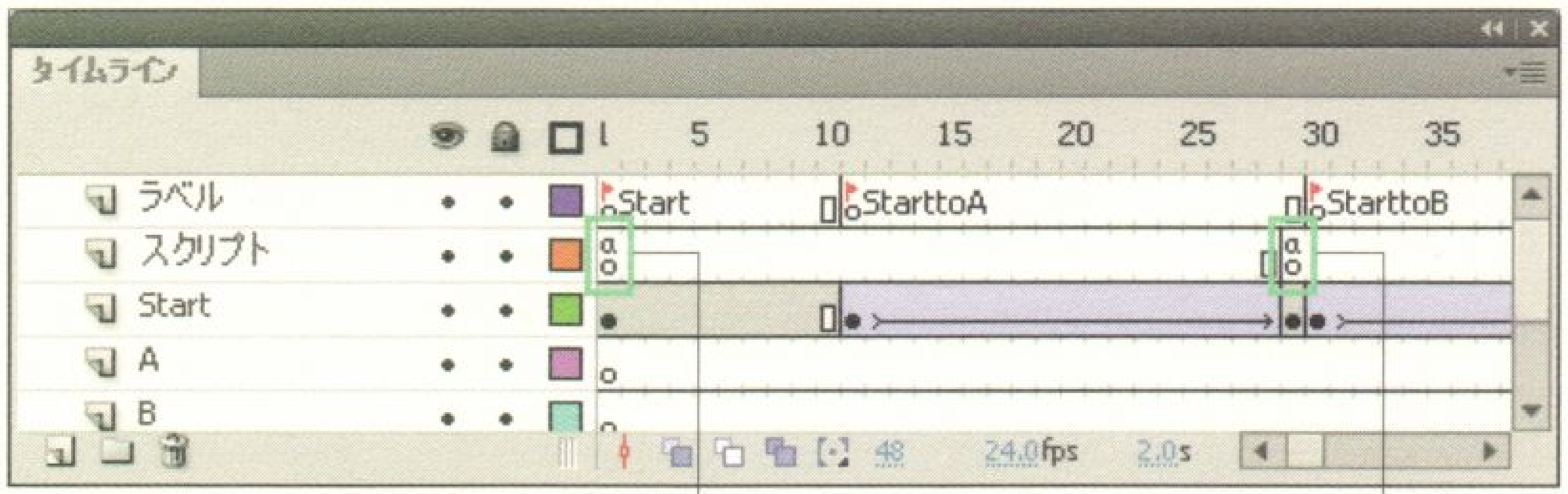


3 後ほどActionScriptでそれぞれのアニメーションを制御するため、1フレーム目に「Start」、それ以外のアニメーションがスタートするフレームには「StarttoA」、「BtoD」という形で『現在地to目的地』という規則を持たせたラベルを付けておきます。
「ラベル」レイヤーでラベルを付けたいフレームを右クリック→［空白キーフレームの挿入］で空白のキーフレームを挿入し、［プロパティ］パネルの「名前」にラベル名を入力します。





4 それぞれのスクロールアニメーションを停止させて、かつ現在表示されている画面（クリックしたボタンに対応している）の情報を持つ変数nowSceneをスクリプトに書いていきます。「スクリプト」レイヤーの次の位置で［アクション］パネルを開き、次のように記述します。



1フレーム目　29フレーム目

1フレーム目

```
stop(); ← ムービークリップの再生を中止
var nowScene:String = "Start"; ←現在表示されている画面の情報を扱う変数nowSceneを宣言し、"Start"を代入
```

スクロールアニメーションが終わるフレーム（例:29フレーム目）

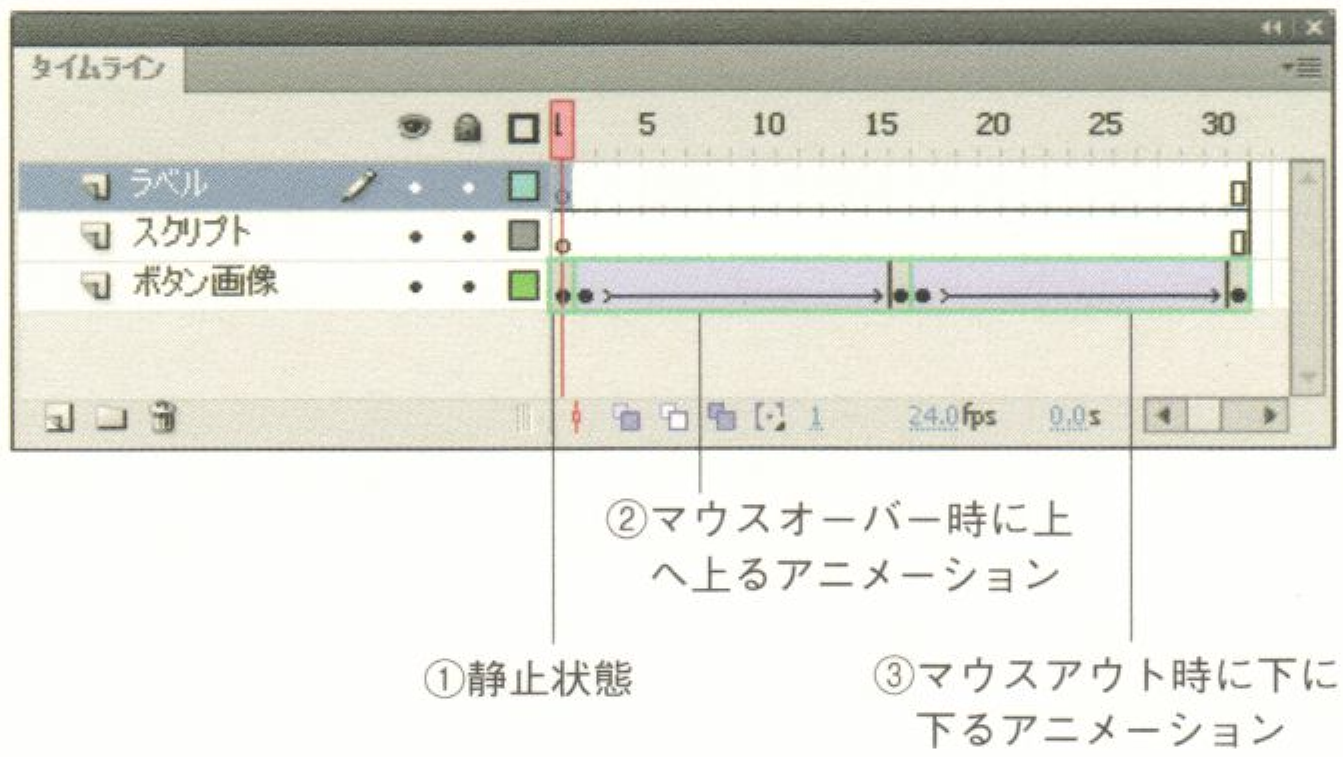
```
stop();
nowScene = "A" ←そのフレームで表示されている画面の情報(Start,A,B,C,D)を変数nowSceneに代入
```

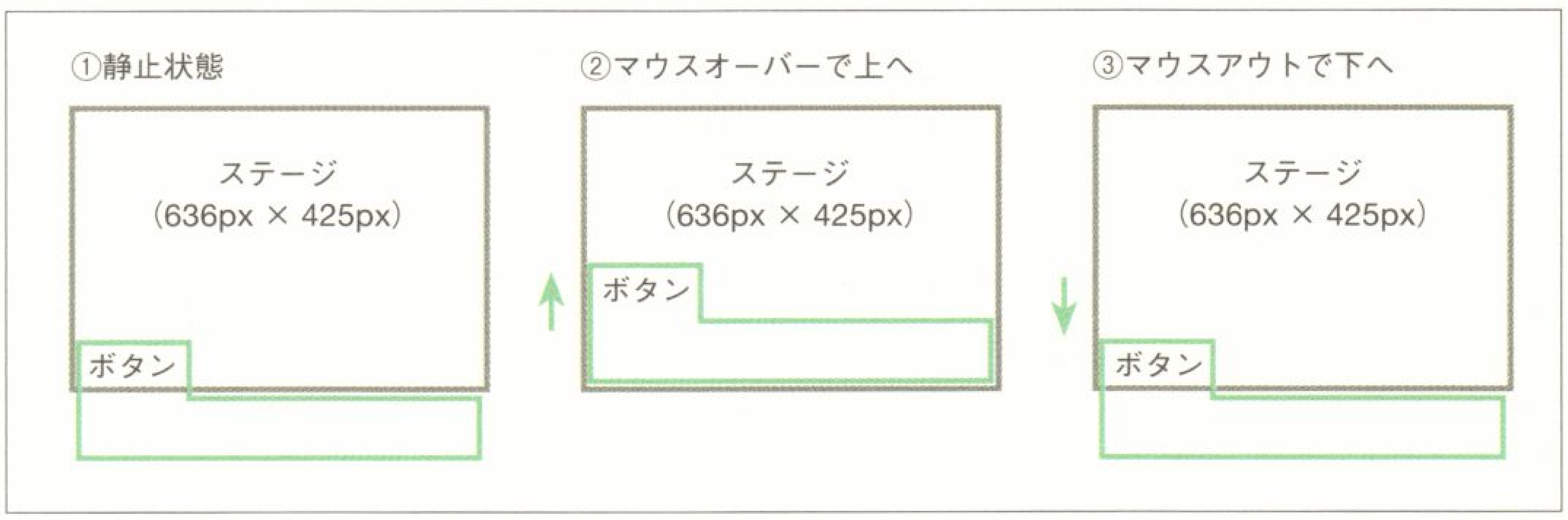
STEP 3 ボタンのアニメーションを作る

次に、ボタンのアニメーションを「btnA(～D)_mc」の中で作ります。
STEP 2同様、アニメーションの制御ができるように、それぞれの動きに対してフラグを付けておきます。

1

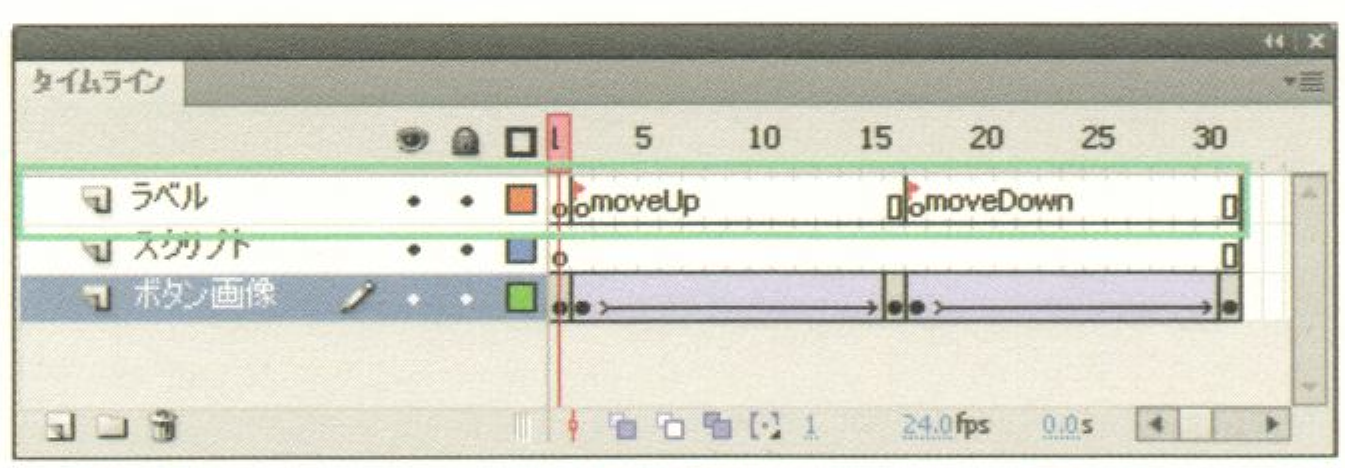
btnA_mcの中のレイヤー構造は右図のようにしておきます。
ボタンの動きのパターンは、①静止状態、②マウスオーバー時に上にせり上がるアニメーション、③マウスアウト時に下に下がるアニメーションの3つのパターンを作ります。
「ボタン画像」レイヤーにbtnA_imageを配置し、1フレーム目は静止している状態、2～16フレーム目で上に上がるアニメーション、17～31フレームで下に下がるアニメーションを作成します。





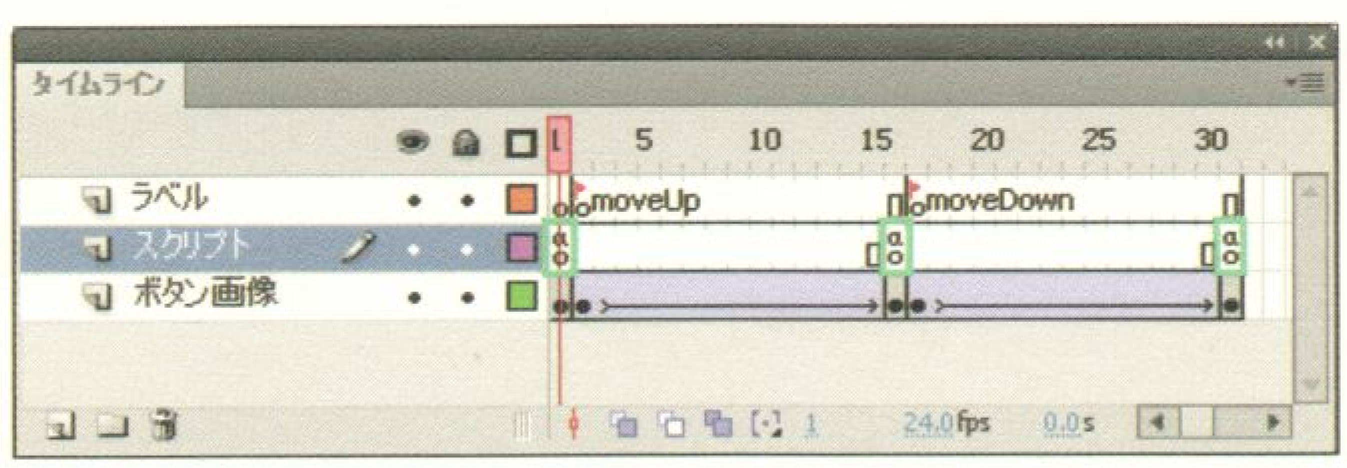
2

あとでスクリプトで制御できるように、作成した2つのアニメーションが開始するフレームにラベルを付けます。
2フレーム目、17フレーム目に空白のキーフレームを挿入し、[プロパティ]パネルの「名前」にそれぞれ「moveUp」、「moveDown」とラベル名を付けておきます。



3

静止している状態と、2つのアニメーションが停止した際にタイムラインを停止させるため、「スクリプト」レイヤーの1フレーム目、16フレーム目、31フレーム目で[アクション]パネルを開き、stop();を書いておきます。
これでbtnA_mcの準備は完了です。同様の作業をbtnB_mc～btnD_mcで行ってください。



1、16、31フレーム目

```
stop(); ⬅ムービークリップの再生を中止
```

STEP 4 ボタンの動きをスクリプトで制御する

いよいよプログラミングに入っていきます。
まずはボタンをマウスオーバー／アウトしたときの動きをActionScriptで制御します。

1 まずはボタンのムービークリップ(btnA_mc～btnD_mc)をボタン化し、さらにその子インスタンスがマウスに反応しない設定を行い、マウスオーバー／アウトしたときのイベントを登録します。
メインタイムラインの「スクリプト」レイヤーの1フレーム目で［アクション］パネルを開き、右のように書きます。
いずれのボタンも、マウスオーバー／アウト時に呼ばれる関数は同じものを使用します。

■イベントリスナ…………… P090へ

btnA_mcの場合

```
//btnA_mcの子インスタンスに対してマウスが反応しないようにする
btnA_mc.mouseChildren = false;

//btnA_mcをボタン化し、オーバー時にマウスカーソルを変化させる
btnA_mc.buttonMode = true;

//ロールオーバー時
btnA_mc.addEventListener(MouseEvent.ROLL_OVER, onBtnOver);
⬆ btnA_mcにマウスオーバーしたら、関数onBtnOverを呼ぶ

//ロールアウト時
btnA_mc.addEventListener(MouseEvent.ROLL_OUT, onBtnOut);
⬆ btnA_mcからマウスアウトしたら、関数onBtnOutを呼ぶ
```

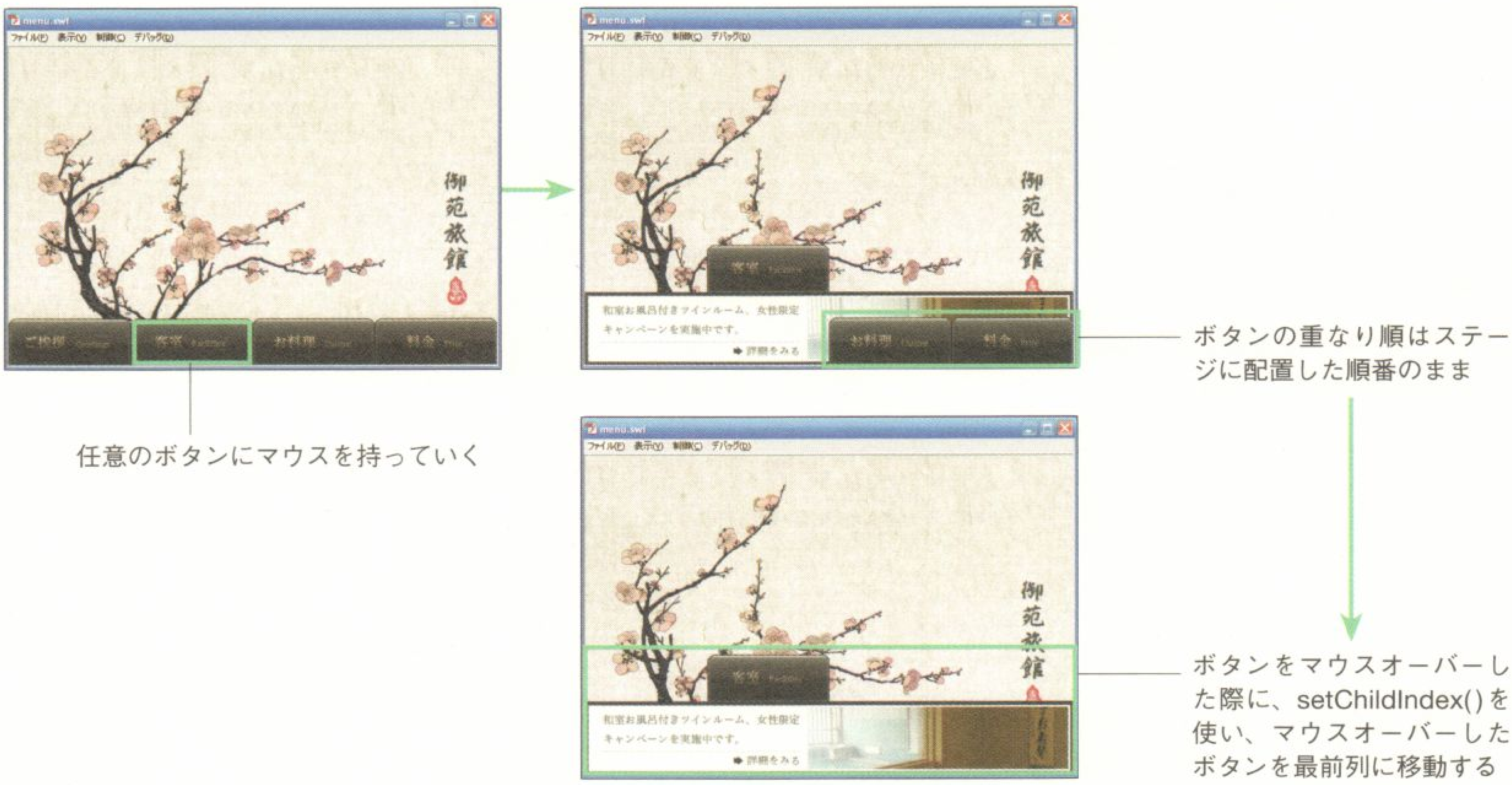
2 次に、ボタンからイベントが発行されたら実行される関数、onBtnOverとonBtnOutを作ります。
ここではbtnA～D_mcで作成した、上る／下るアニメーションを再生したいので、gotoAndPlay( )メソッドを使い、マウスオーバーのときはラベル名「moveUp」、マウスアウトのときは「moveDown」のフレームから再生ヘッドがスタートするようにスクリプトを書きます。

```
//マウスオーバーしたら実行される
function onBtnOver(event){
    event.target.gotoAndPlay("moveUp");   ⬅ event.targetで、イベントを送出したオブジェクトを特定できる
}

//マウスアウトしたら実行される
function onBtnOut(event){
    event.target.gotoAndPlay("moveDown");   ⬅ event.targetで、イベントを送出したオブジェクトを特定できる
}
```

■タイムラインを制御する……………… P080へ

3 この時点でムービープレビューしてみると分かりますが、このままだとボタンの重なり順がステージ上で配置したままになっていて、他のボタンの説明文が見れません。そこでマウスオーバーされたボタンが最前面に配置するために、setChildIndex( )メソッドを使ってボタンの重なり順を変更します。




```
//マウスオーバーしたら実行される
function onBtnOver(event){
    setChildIndex(event.target, numChildren - 1);   //マウスオーバーされたムービークリップを最前面に配置する
    ⬆setChildIndex(重なり順を変更したいインスタンス,変更後の重なり順) ※numChildrenはオブジェクトの子インスタンスの数を返す
    event.target.gotoAndPlay("moveUp");
}
```

STEP 5 メインイメージの動きをスクリプトで制御する

ボタンのマウスオーバー／アウトが動くようになったら、次はボタンをクリックしてメインイメージのアニメーションを制御する仕組みを作ります。ポイントは、メインイメージが止まっている位置を、変数とフラグ名を使って管理することです。

1 まずはSTEP 4同様、ボタンがクリックされたときのイベントを登録します。今回は、クリックされたときに実行される関数はクリックされたボタンによって動作が異なるので、それぞれ別のものにしておきます。

```
//「ご挨拶」ボタン(btnA_mc)クリック時
btnA_mc.addEventListener(MouseEvent.CLICK, onBtnAClick);

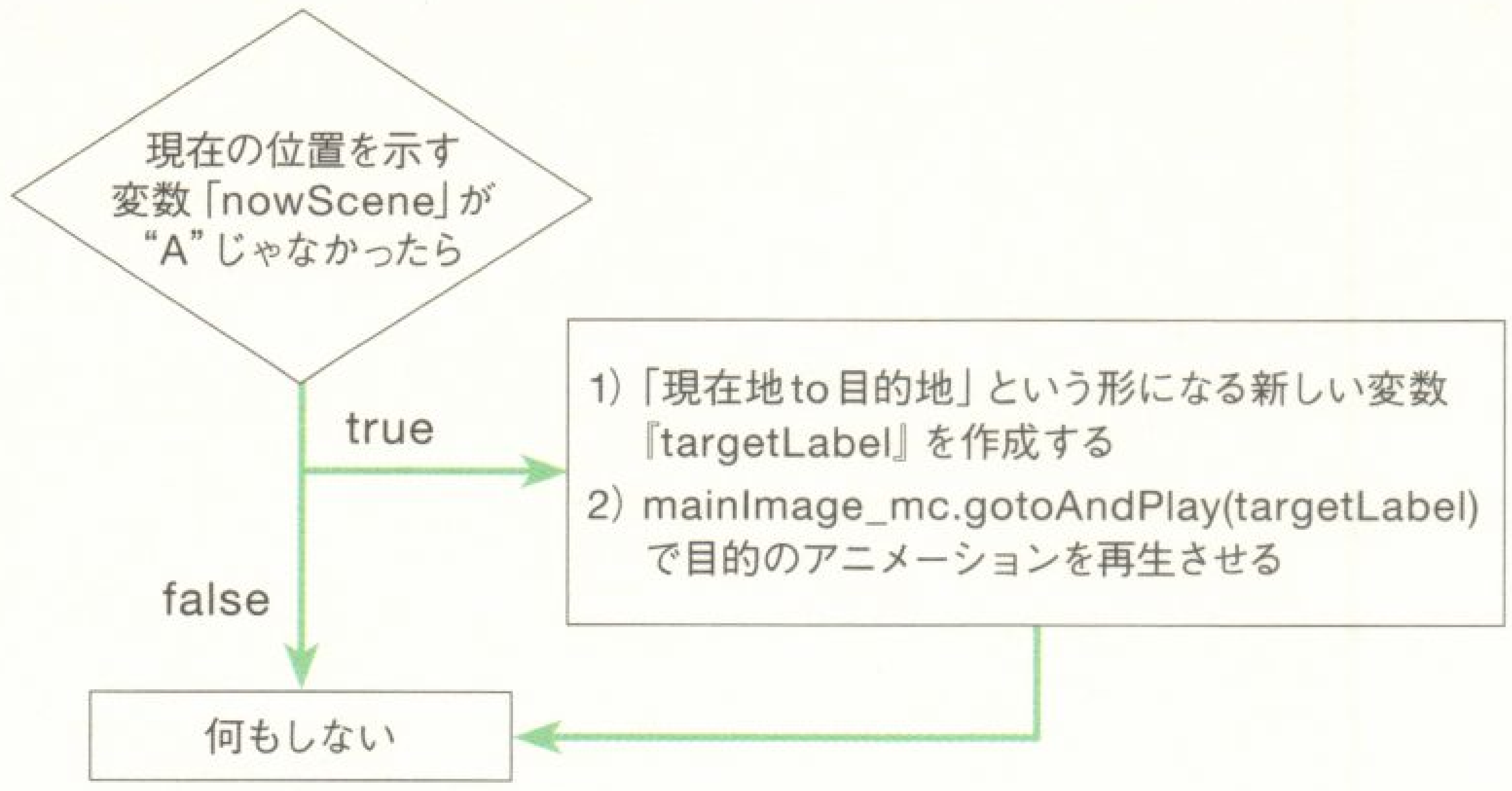
//「客室」ボタン(btnB_mc)クリック時
btnB_mc.addEventListener(MouseEvent.CLICK, onBtnBClick);

//「お料理」ボタン(btnC_mc)クリック時
btnC_mc.addEventListener(MouseEvent.CLICK, onBtnCClick);

//「料金」ボタン(btnD_mc)クリック時
btnD_mc.addEventListener(MouseEvent.CLICK, onBtnDClick);
```

2 次に関数を作ります。ボタンをクリックした際に、mainImage_mc内のどのアニメーションを再生させるかを決めるために、変数nowSceneとif文による条件分岐を使って、図のように考えます。

■ 変数 ……………………………P094へ
■ if ……………………………………P107へ
■ function ……………………………P086へ




```
//btnA_mcがクリックされたら実行される
function onBtnAClick(event){
    if(mainImage_mc.nowScene != "A"){
        var targetLabel= mainImage_mc.nowScene + "toA";   ⬅「現在地toA」という形の文字列ができる 例) start to A
        mainImage_mc.gotoAndPlay(targetLabel);   ⬅ mainImage_mc内のラベル名をtargetLabelを使って指定できる
    }
}
```

3 ここで一度動きを見てみましょう。メニューバーの［制御→ムービープレビュー］でムービープレビューします。ボタンのマウスオーバー／アウトでボタンが上下し、クリックで後ろのメインイメージが「現在地→目的地」の形でスクロールしていればＯＫです。



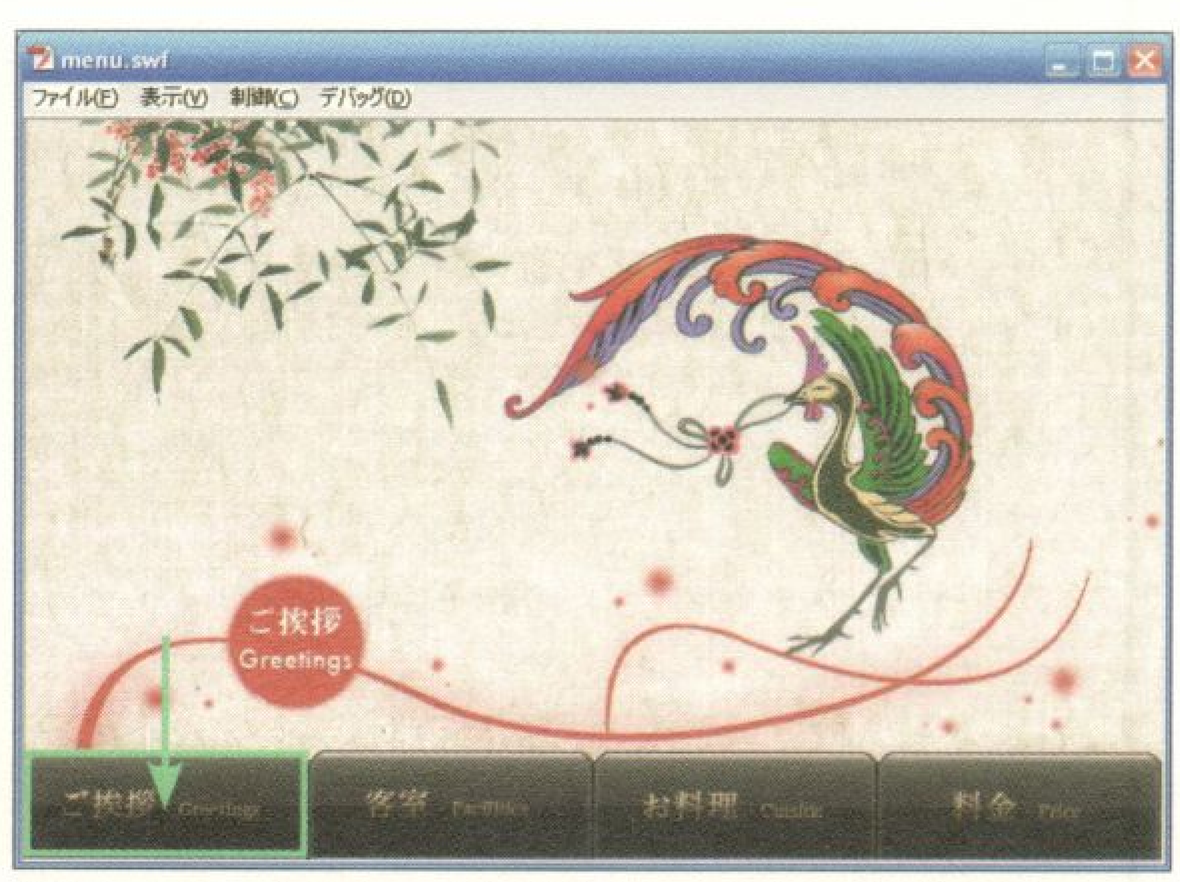
パブリッシュした状態



「ご挨拶」ボタンにマウスカーソルを合わせると、ボタンがせり上る



ボタンをクリックすると、後ろのメインイメージが「ご挨拶」画面にスクロールする



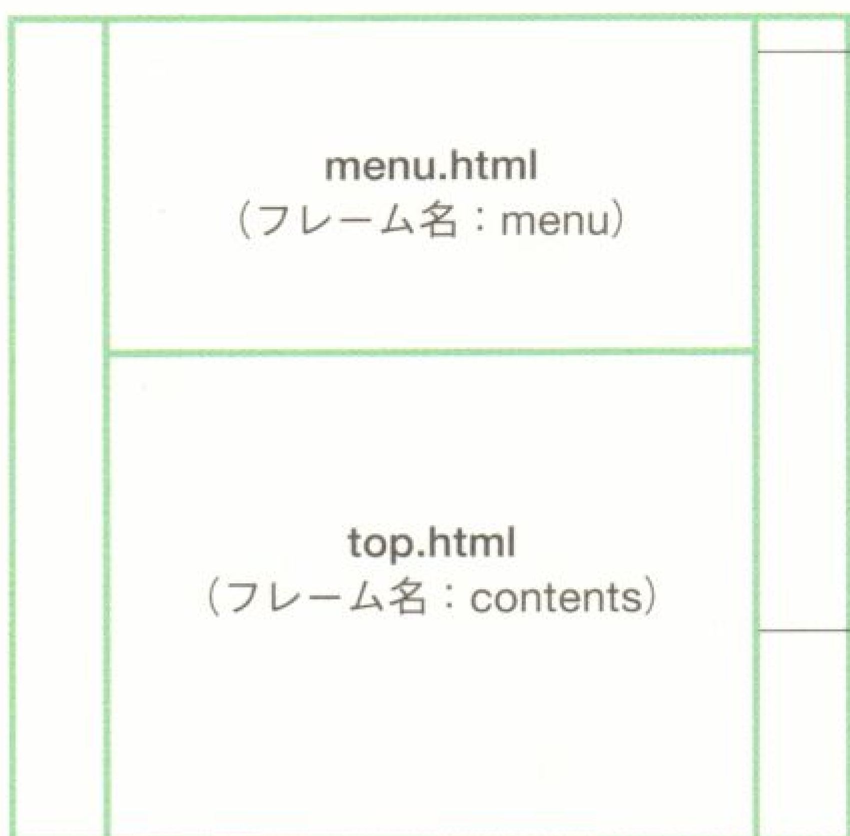
ボタンからマウスカーソルを外すと、ボタンは下に下る

STEP 6 HTMLページを切り替える仕組みを作る

いよいよ最後の仕上げ、ボタンクリックでHTMLページを切り替える仕組みを作ります。ここではnavigateToURLというメソッドを使ってフレーム内のHTMLページを切り替えます。

1 今回はフレームを使ってページの切り替えを行うので、各フレームが読み込まれるframesetが書いてあるベースとなるHTMLページ(index.html)、メニューのswfを読み込むページ(menu.html)、そしてメニューによって切り替わるページを用意します。

メニューとコンテンツは、それぞれ「menu」、「contents」というフレーム名をindex.htmlのframesetの中で付けておきます。menu.htmlは、今回作成しているFlashファイルから書き出したものを使用します。



menu.html：Flashから書きだしたHTMLファイル

index.html

ソース(抜粋)

```
<frameset rows= "425,*" >
  <frame src= "menu.html" name= "menu.html" >
  <frame src= "top.html" name= "contents" >
</frameset>
```

切り替わるHTMLページ： **top.html**(トップページ)
greetings.html(「ご挨拶」ページ)
facilities.html(「客室」ページ)
cuisine.html(「お料理」ページ)
price.html(「料金」ページ)

---

2 HTML側の準備ができたら、ボタンをクリックした際に実行される関数onBtnA～DClickにページを切り替えるスクリプトを追加します。

navigateToURL( )を使い、btnA～D_mcに対し、対応するHTMLファイルをフレーム名「contents」のページに読み込ませまるようにします。

```
//btnA_mcがクリックされたら実行される
function onBtnAClick(event){
  if(mainImage_mc.nowScene != "A"){
    var targetLabel= mainImage_mc.nowScene + "toA";
    mainImage_mc.gotoAndPlay(targetLabel);
    navigateToURL(new URLRequest("greetings.html"),"contents");
    ⬆ greeting.htmlをcontentsという名前のフレームに読み込む
  }
}
```

■ navigateToURL( ) ························ P167へ

これで完成です。ファイルをパブリッシュして、動作を確認してください。

注意点としては、ローカルにあるswfファイルからnavigateToURLを使ってコンテンツを呼び出す際、コンテンツがサーバ上にある場合はセキュリティエラーが出ることがあります。この点については、コラムで解説します。

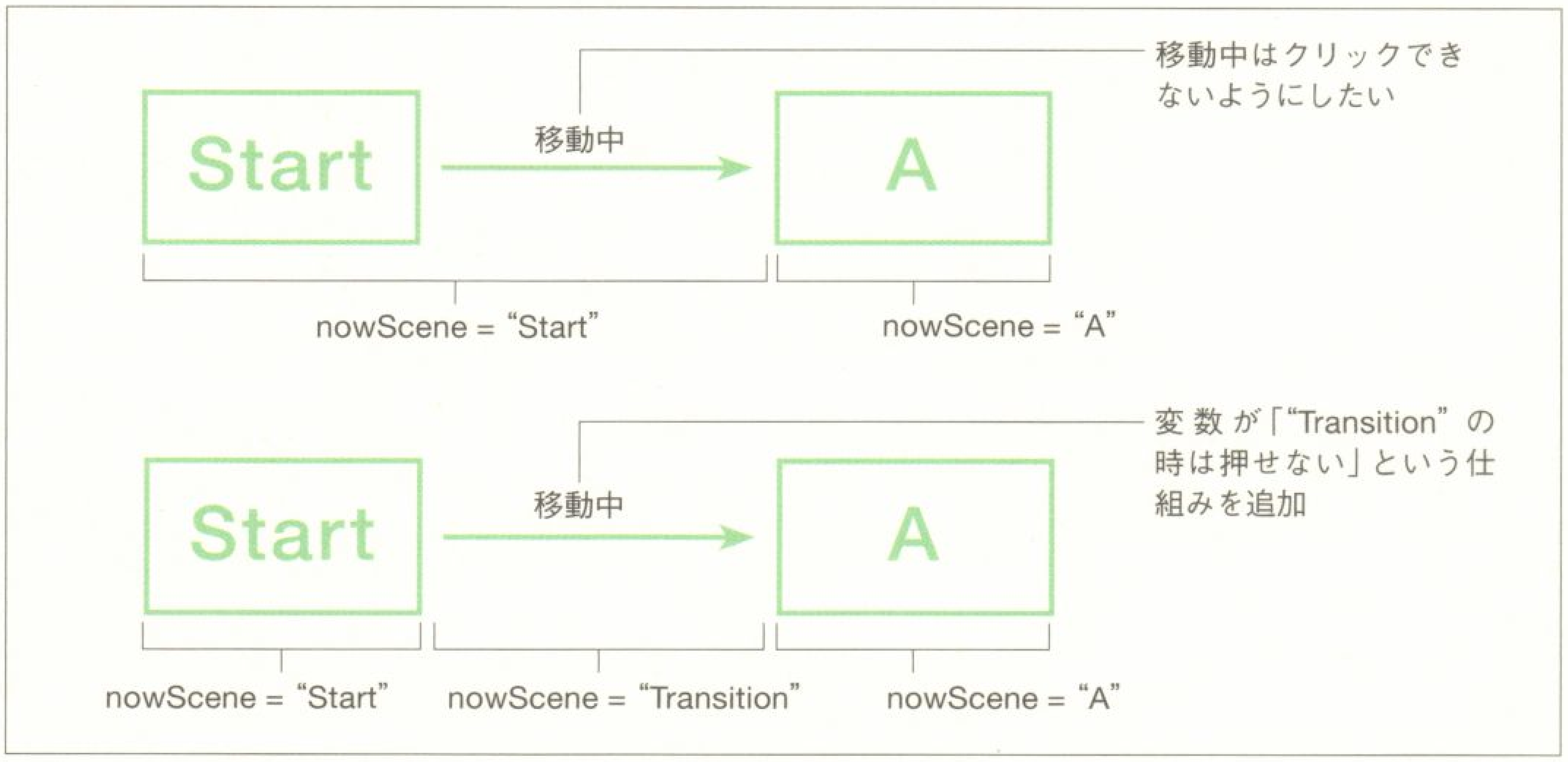
応用

Ex.1 遷移中にボタンが効かないようにする

さらに完成度を上げるために、ボタンの２重押しができないような仕組みを組み込んでみます。
今回使った変数の仕組みに少し手を加えるだけで実装が可能ですので、是非チャレンジしてみてください。

1 STEP 6まで完成したサンプルを動かしてみると分かりますが、一度ボタンをクリックしてメインイメージがスクロールしている途中にもう一度ボタンをクリックすると、スクロールのアニメーションがまた最初からスタートしてしまいます。このままでも良いのですが、ディテールを上げるため、メインイメージがスクロールしている間は、ボタンをクリックしてもまたスクロールが最初から始まらない仕組み(ボタンロック機能)を考えてみます。

2 図は、ページ読み込み後、btnA_mcをクリックした際の変数nowSceneの値の動きを表しています。
図の上側はこれまで作成したサンプルの状態です。ここでは一度ボタンをクリックしてメインイメージが移動している状態のnowSceneの値は"Start"のままなので、移動中に再度ボタンをクリックすると、そこで実行される関数onBtnAClickの中の「nowSceneの値がAではなかったら」という条件に当てはまり、またアニメーションが実行されてしまいます。
そこで図の下側のように、ボタンをクリックした際に、nowSceneの値に移動中を表す"Transition"をいう新しい値を代入し、onBtnAClick内の条件を、「nowSceneがAまたはTransitionでなければ」と変えることでスクロール中にボタンが効かない仕組みを実装します。



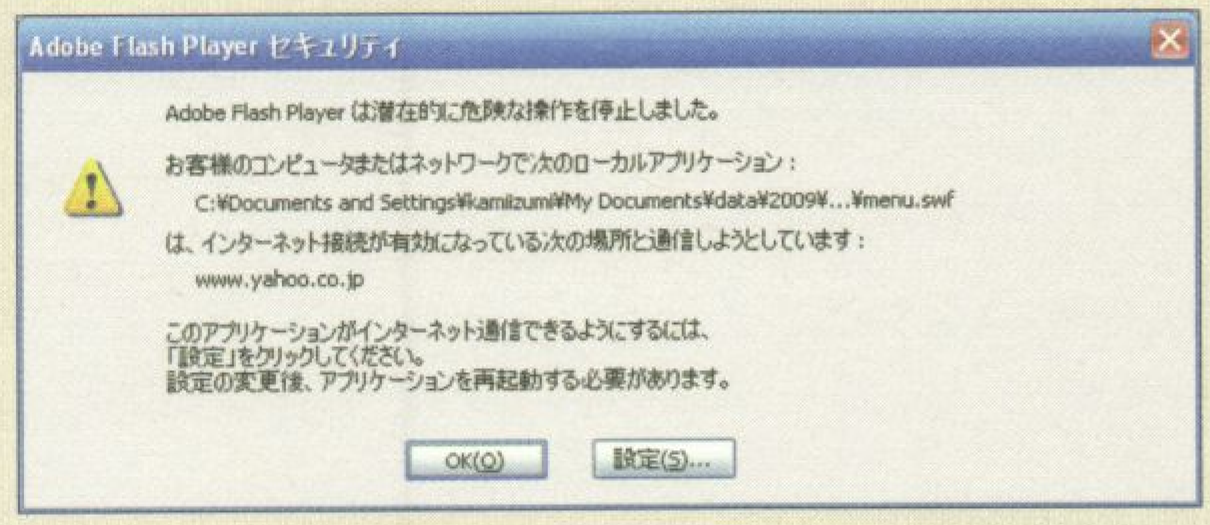
3 では、実際にコードを書いてみましょう。関数onBtnA（～D）Clickの中に、下記の部分を追加します。ムービープレビューでボタンを２度押しても、メインイメージのスクロールアニメーションがまた始めから再生されなければ、完成です。

```
//btnA_mcがクリックされたら実行される
function onBtnAClick(event){
  if(mainImage_mc.nowScene != "A" && mainImage_mc.nowScene != "Transition"){
  ⬆ 条件に「かつmainImage_mcの変数nowSceneがTransitionではなかったら」を追加
    var targetLabel= mainImage_mc.nowScene + "toA";
    mainImage_mc.gotoAndPlay(targetLabel);
    mainImage_mc.nowScene = "Transition";  ⬅ mainImage_mcの変数nowSceneにTransitionを代入
    navigateToURL(new URLRequest("greetings.html"),"contents");
  }
}
```

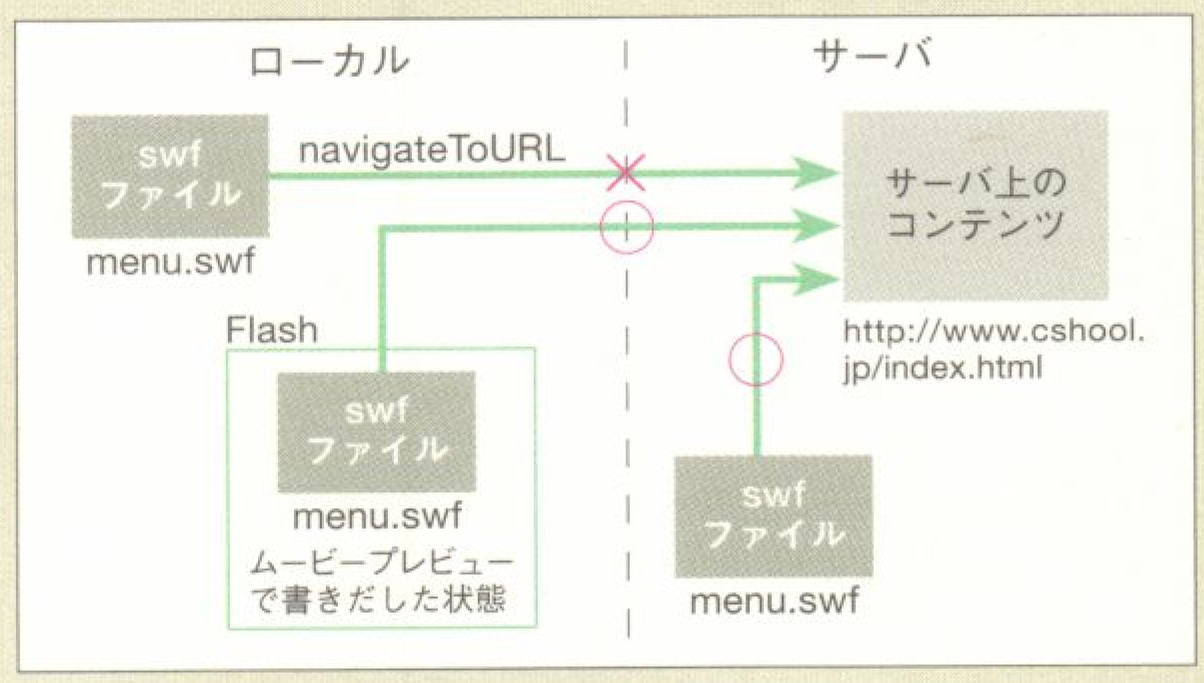
COLUMN

ローカルにあるswfファイルからインターネット上のコンテンツにアクセスする際の注意点

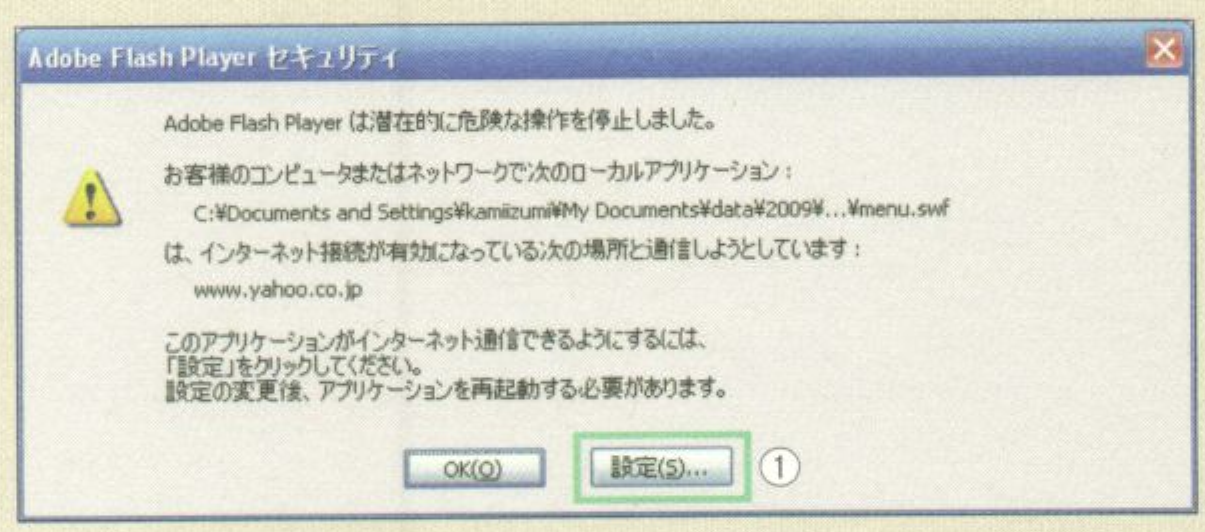
ローカルにあるswfファイルからnavigateToURL( )を使って、インターネット上にあるコンテンツを呼び出す場合、このようなエラーが出ることがあります。ここでは、その原因と対処方法について解説します。



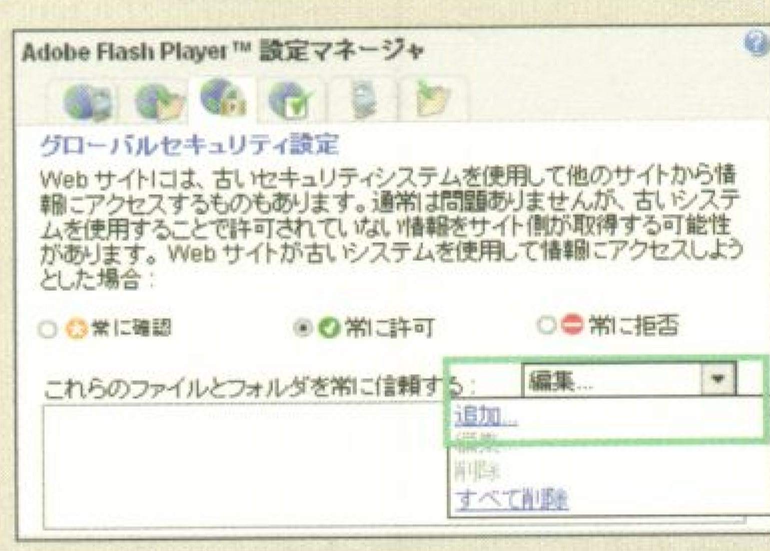
このエラーは、ローカルにあるswfファイルがインターネット上のコンテンツに勝手にアクセスできないようにする、Flashのセキュリティ機能によるものです。swfファイルがサーバ上にある場合、またはローカル上でもFlashからムービープレビューした状態のファイルからアクセスした場合は、この現象は発生しません。
エラーが出る／出ないケースをまとめると、右図のようになります。



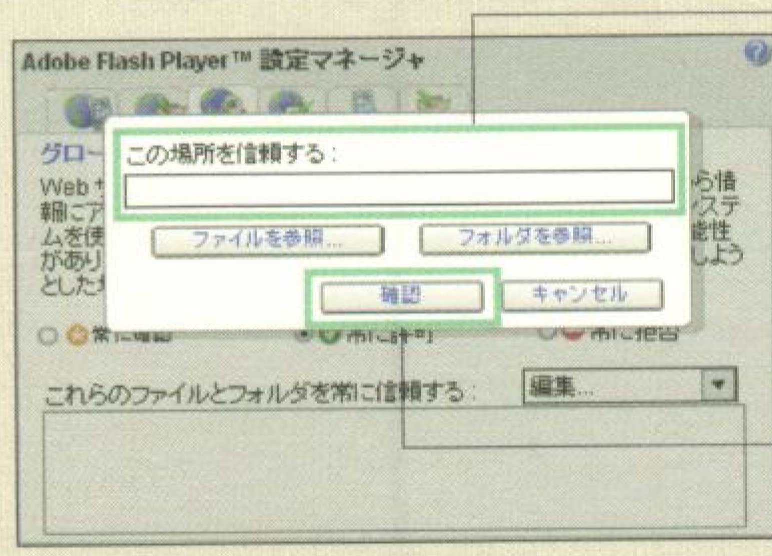
このエラーを回避するには、ローカルにあるswfファイルを指定し、そのファイルからインターネット上のコンテンツへのアクセスを許可する設定をする必要があります。
まず、エラーメッセージの画面から、［設定］ボタンをクリックし(①)、Adobeのサイトにある「Adobe Flash Player設定マネージャ」のページを開きます。



次に、設定マネージャの［編集 → 追加］(②)から、アクセス許可をしたいファイルを指定し(③)、［確認］をクリックし、設定を完了します(④)。(コンテンツ作成者のためのグローバルセキュリティ設定：http://www.macromedia.com/support/documentation/jp/flashplayer/help/settings_manager04a.html)



②［編集 → 追加］をクリック



③外部URLを呼び出すswfファイルの場所を指定する（ファイル名、またはファイルのあるフォルダ名でも可）

④［確認］をクリック

Flashにおけるこのようなセキュリティポリシーは、URLLoaderを使ってインターネット上のコンテンツにアクセスしようとした場合でも同様に適用できます。

■ URLLoader ………………………………… P162へ

PRACTICE 3

マウスインタラクションを作る①

表現力：★★★☆☆　プログラミング力：★★☆☆☆

Webサイトの中で、何か少しだけ演出を加えたいと思ったことはないでしょうか。例えば、ティザーサイトなどコンテンツ内容が少ないときに少し、もの寂しく見えてしまうことがあるでしょう。何か1アクションを入れてみたいというときのために、このサンプルを紹介します。ここでは、マウスの位置に合わせて、カラフルな円のアニメーションを表示する表現を練習しましょう。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

- function…… P086へ
- ENTER_FRAME …… P121へ
- マウスの位置を取得する…… P148へ
- addChild …… P124へ
- Math.random …… P130へ

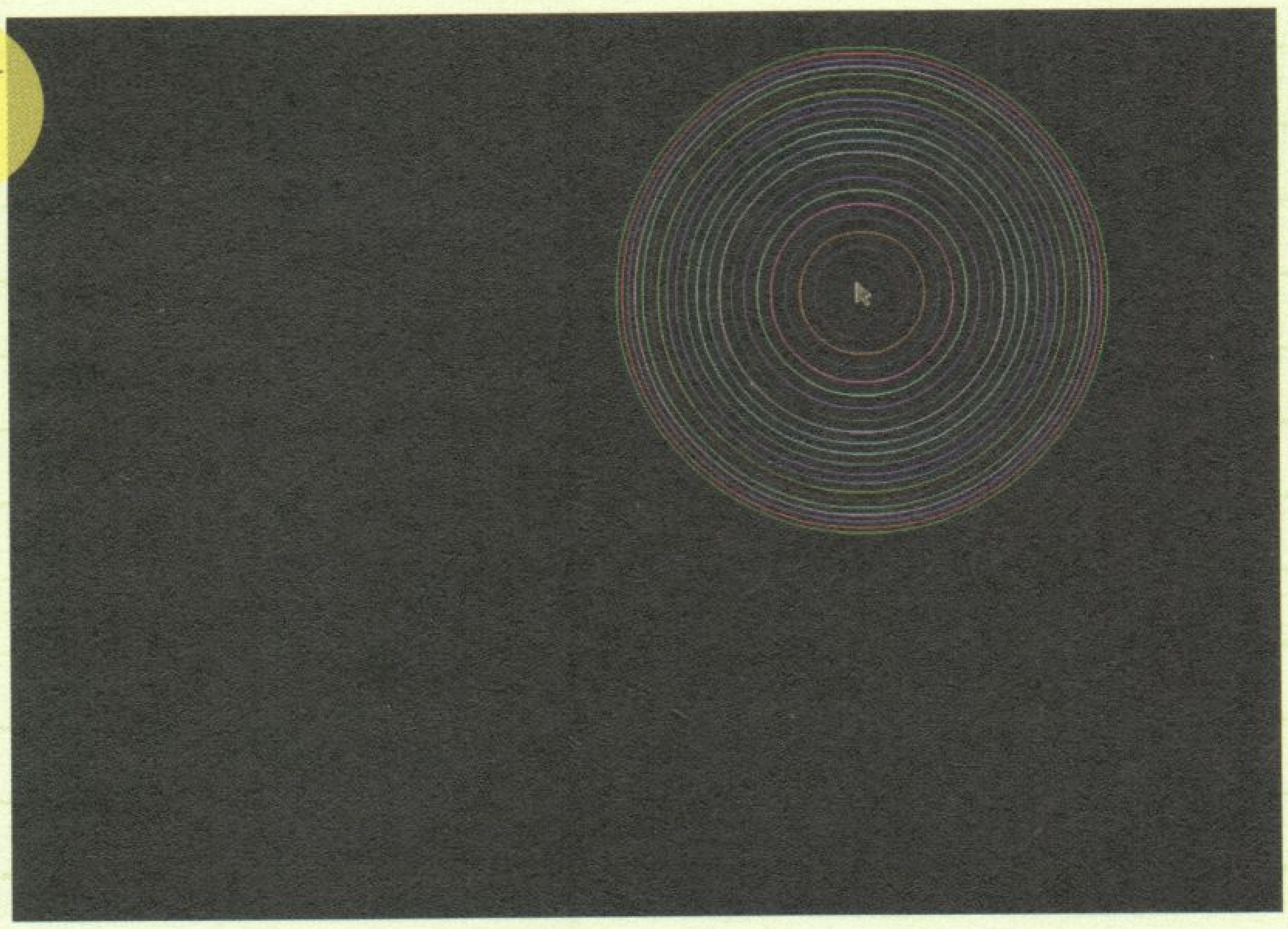
このサンプルで学ぶ応用技術

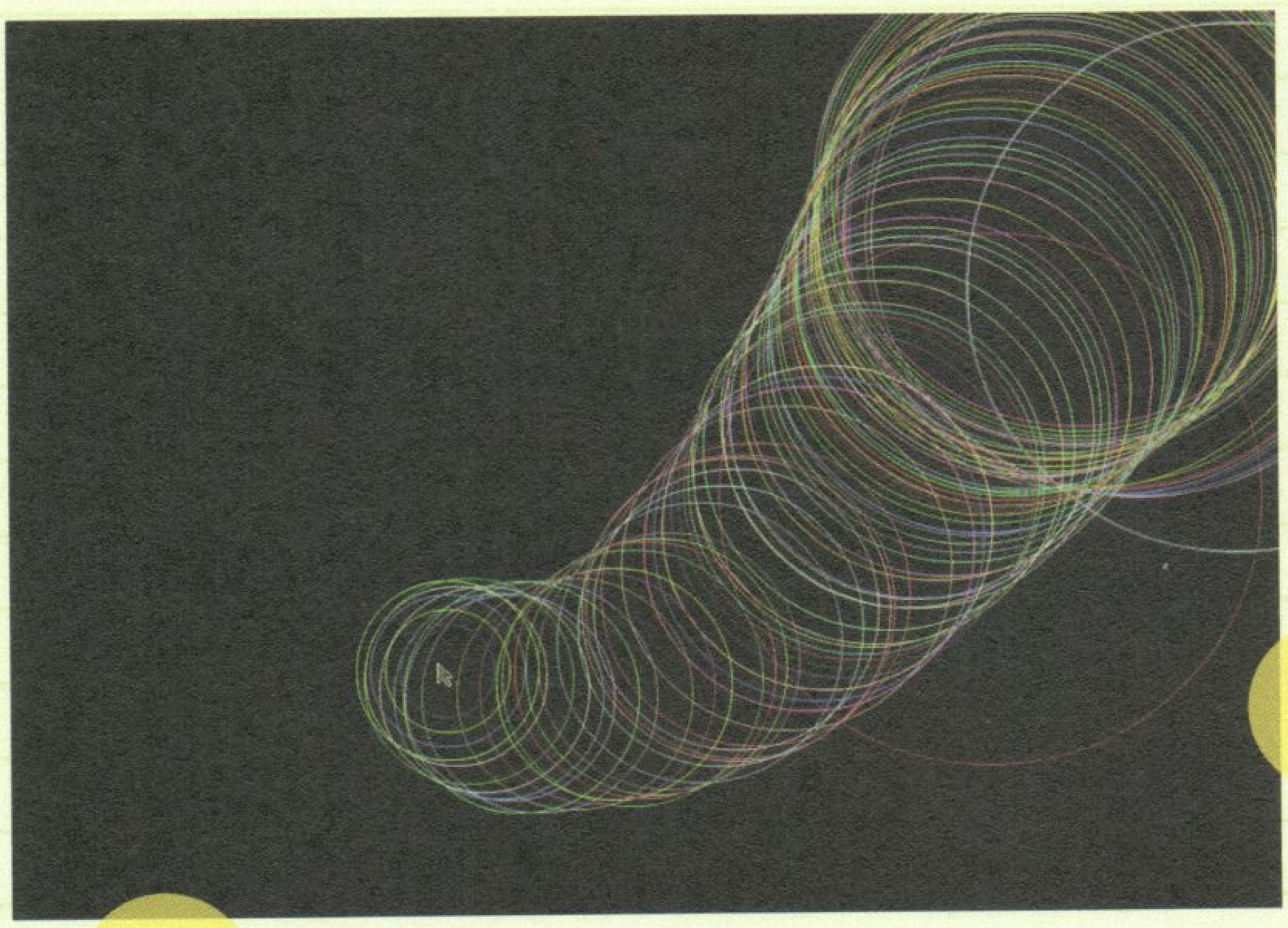
このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- addChildを利用して、ライブラリにあるムービークリップ(ripple)をステージ上に配置
- functionの利用方法
- ENTER_FRAMEを利用して、同じ処理を繰り返す
- ランダムな色のパターンを作る

サンプルの説明

POINT 1





POINT 2

POINT 3

POINT

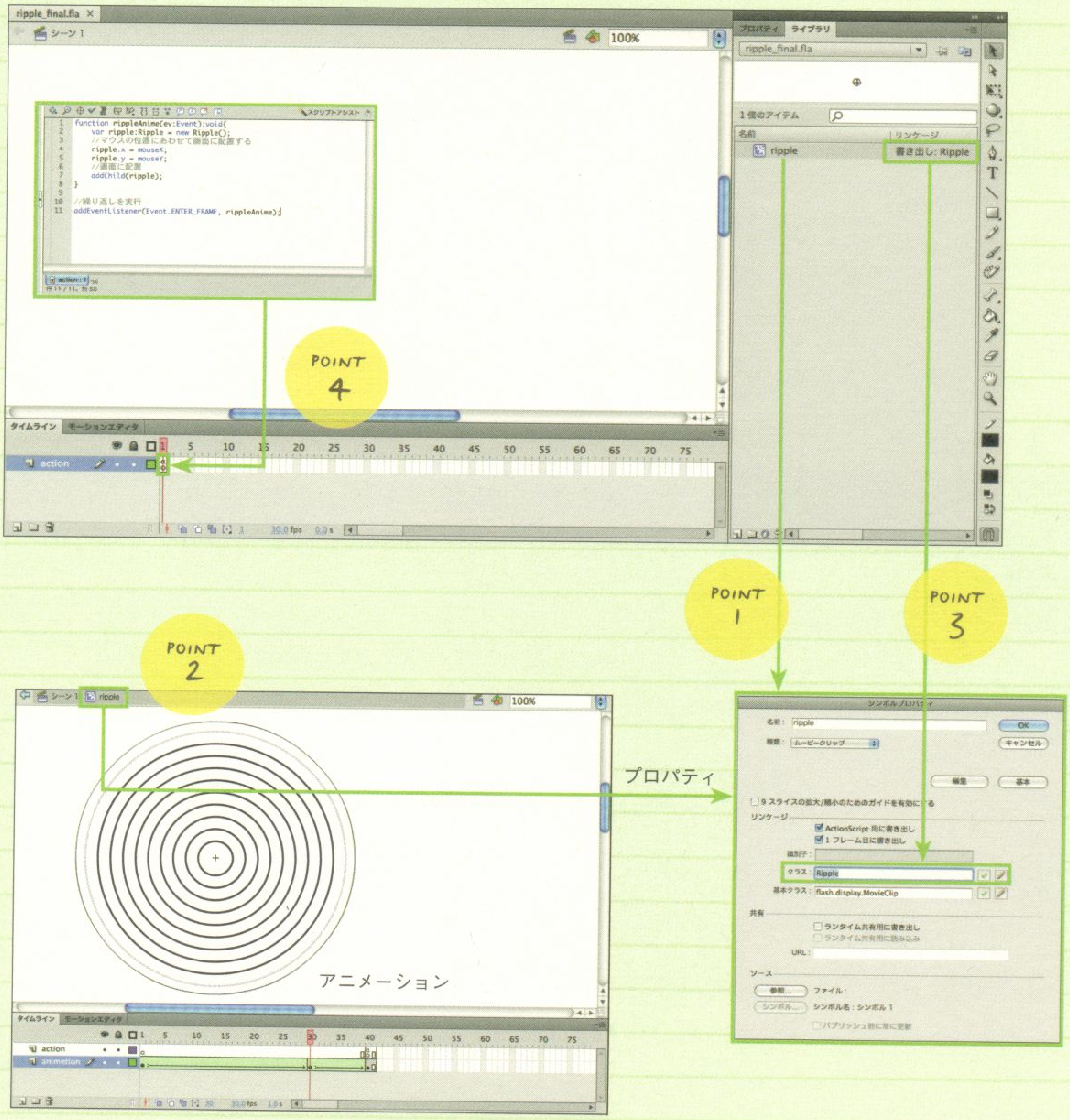
1 画面上のマウスの位置から、カラフルな円が広がっていくアニメーションが表示されます。

2 画面上のマウスの位置を取得して、円のアニメーションがマウスに追従します。

3 円は一定の大きさまで広がると消え、繰り返し円が広がっていくアニメーションが表示されます。

サンプルの構造

ステージ



POINT

1 ActionScriptでアニメーションを入れるRipple（ムービークリップ）を用意します。

2 Ripple（ムービークリップ）の中に、中心（基準点）から外に広がる円のアニメーションを作成します。

3 Ripple（ムービークリップ）を画面上に貼り付けられるよう、リンケージの書き出しをします。

4 メインタイムラインで、Ripple（ムービークリップ）をマウスの位置に動的に貼り付けるActionScriptを書きます。

STEP 1 functionを利用する

今回のサンプルは、画面上のマウスの位置にムービークリップを何度も配置する仕組みを用意します。このように何度も同じ動きを繰り返し実行するときに必須なのがこのfunctionです。

1

まずは［アクション］パネルを開いて、functionを作ります。
メインタイムライン上にActionScriptを書いていきます。
rippleAnimeという functionを作りましょう。

```
function rippleAnime():void{
    trace("rippleAnimeが呼ばれました。");
}
```

■ function…………………………………… P086へ

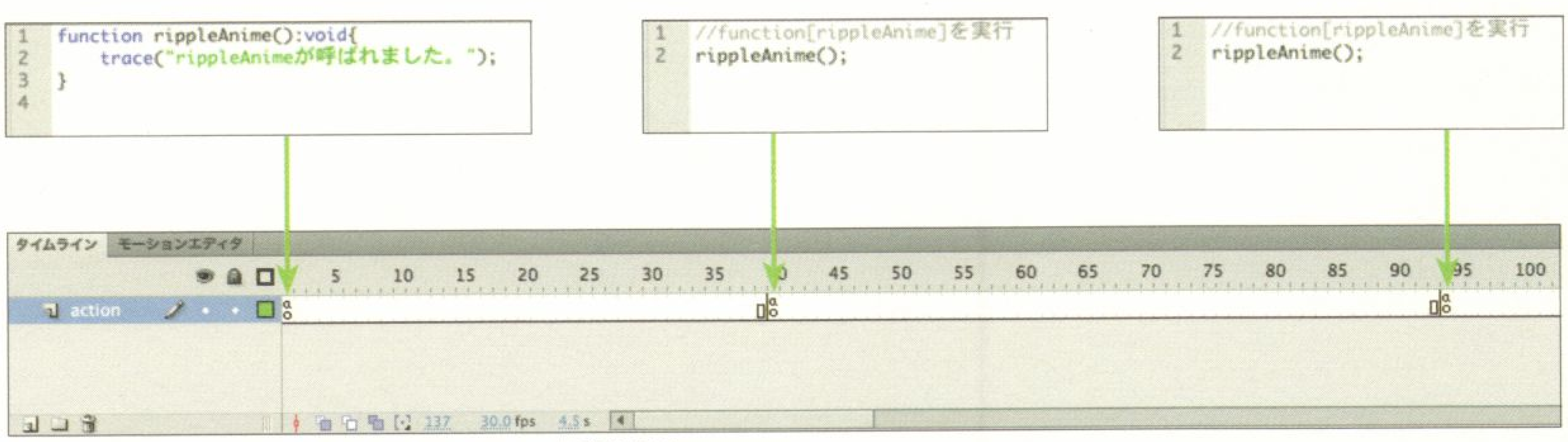
2

functionを実行してみます。
実行の仕方は、この1行のみです。

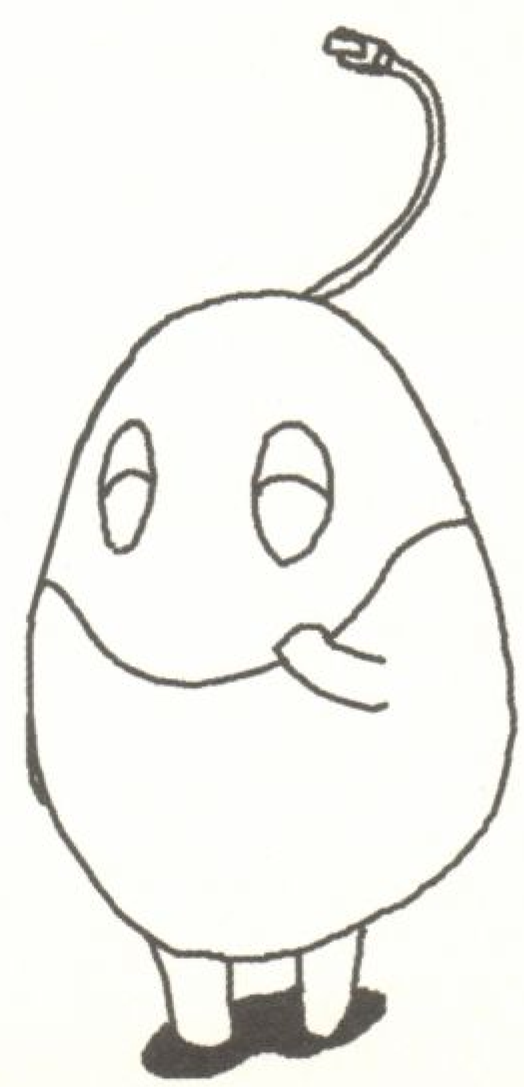
```
//function[rippleAnime]を実行
rippleAnime();
```

3

functionを書き込んだフレーム以降であれば、いつでもこの1行で呼び出すことができます。
実行タイミングを複数箇所用意したとしても、このfunction内を変更するだけで、すべての実行時に適用されます。
試しに、タイムラインで動作するサンプルを用意しました(ripple_function.fla)。



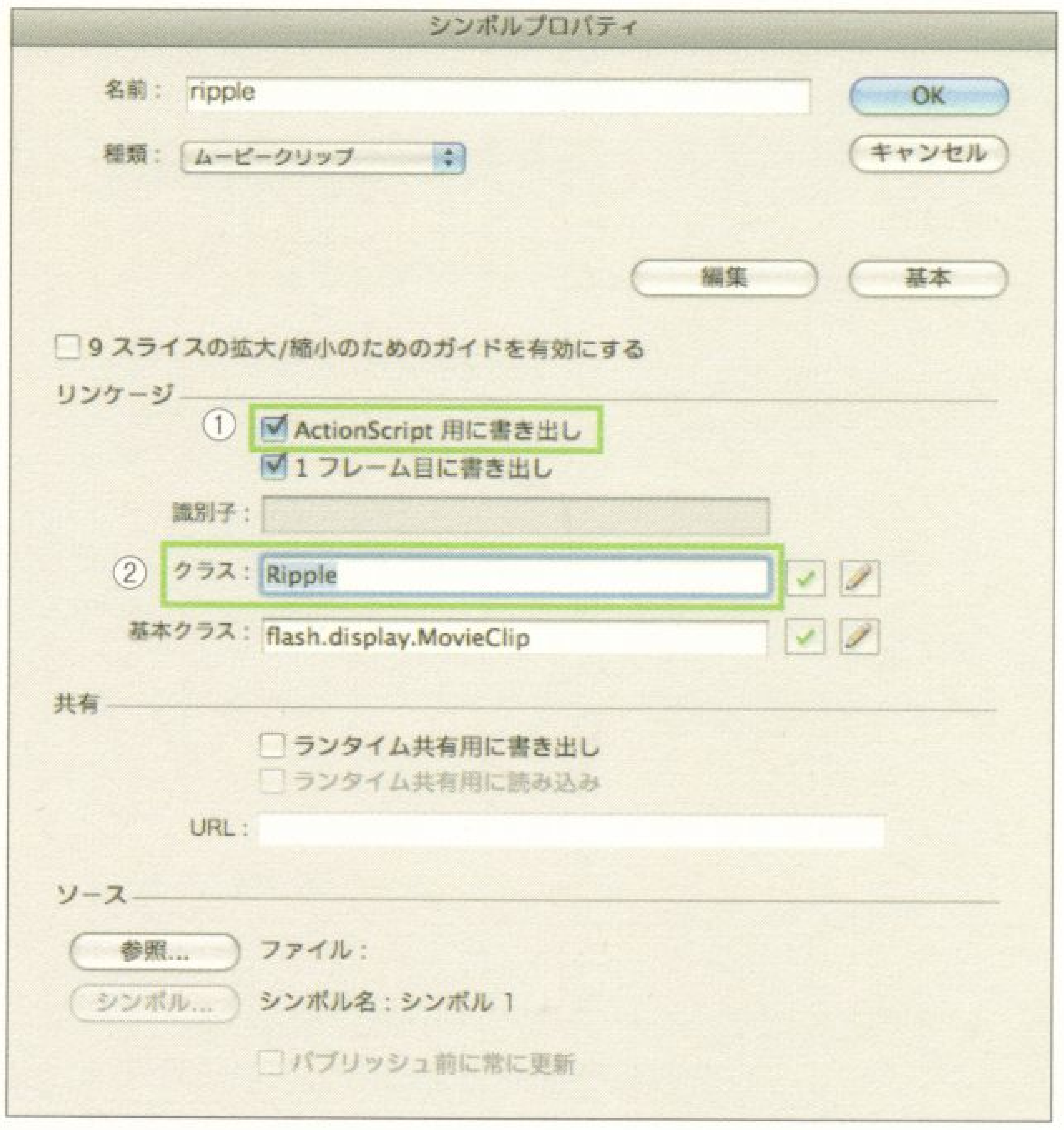
1フレーム目のフレームアクションにfunctionを書き込む。
すると、それ以降のフレームであれば、いつでもfunctionを実行できる。



STEP 2 円のアニメーションを作る

円が中心から広がるアニメーションを作ります。

1 ステージ上に黒い線だけの円を描きます。
ムービークリップに変換し、基準点を中心にします。
さらに右図のように、[ActionScript用に書き出し]にチェックを入れて(①)、クラスを[Ripple]にします(②)。

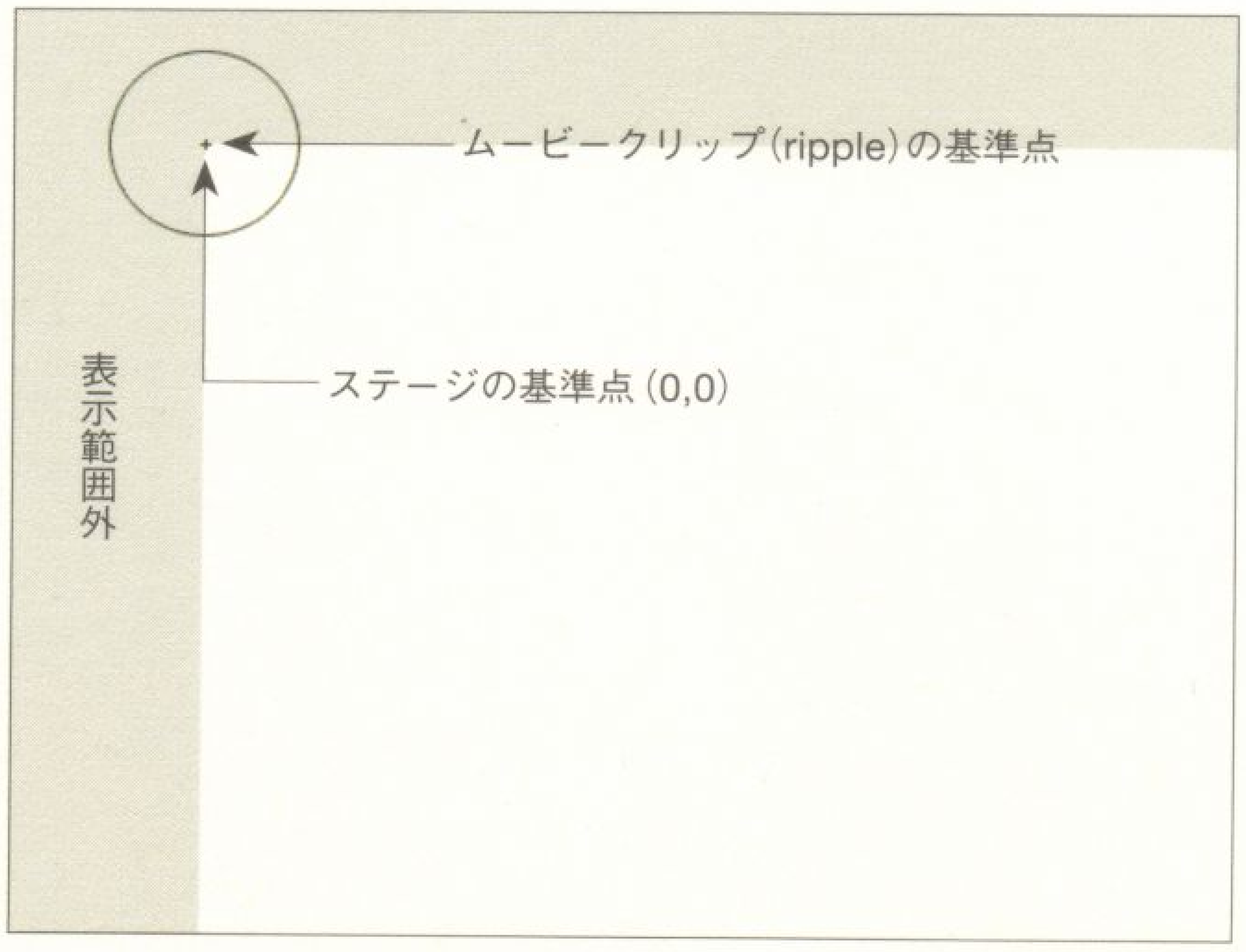


2 先ほど書いたfunction(motion)の中にaddChildを追加します。実行するとステージに配置されます。

■ addChild ………………………… P124へ

```
var ripple:Ripple = new Ripple();
//画面に配置
addChild(ripple);
```

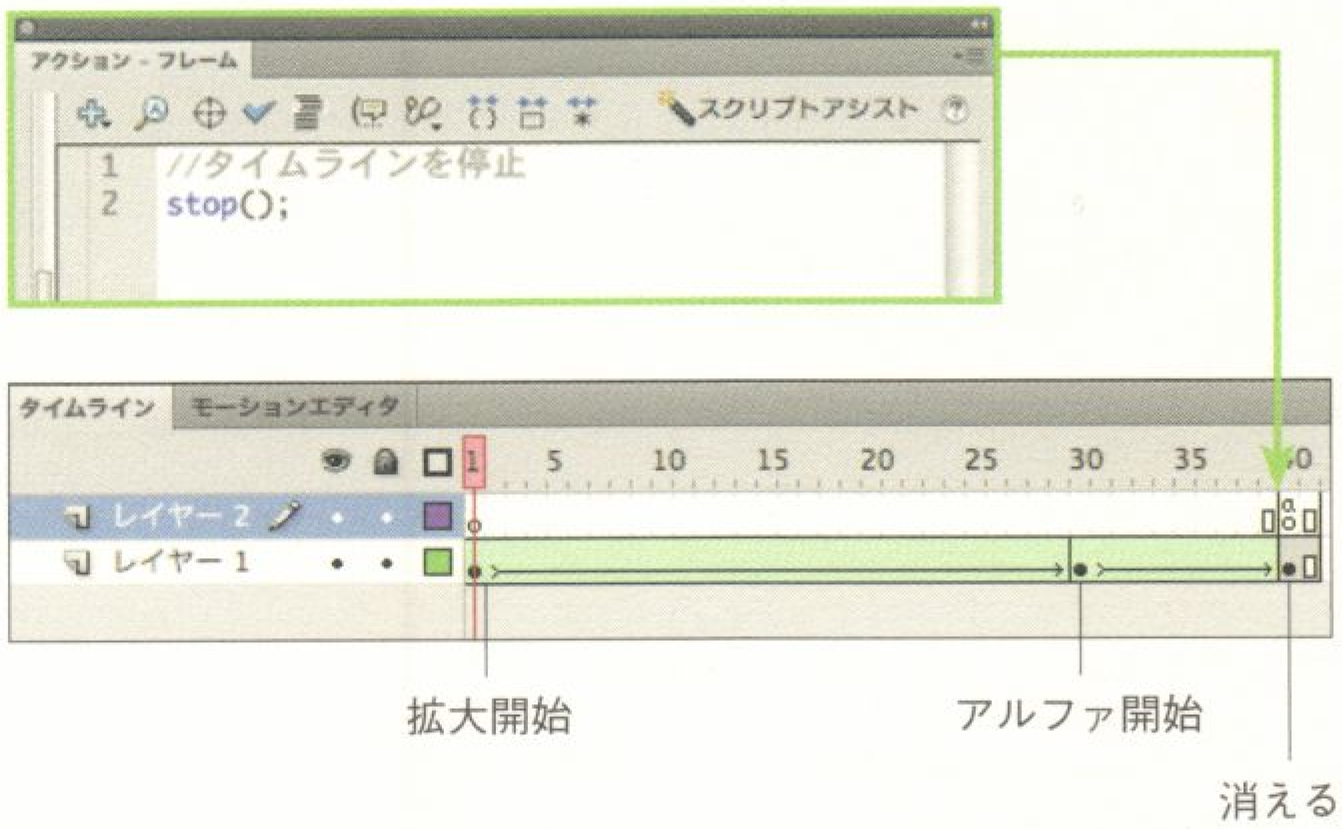
3 実行すると画面左上に丸(1/4)が表示されるはずです。
addChildで画面に配置した場合、位置の指定をしていないと、ステージ上の基準点(0,0)画面左上に配置されます。ムービークリップ(ripple)の基準点は、円の中心にしてあるので、ムービークリップ(ripple)の右下のみが見えている状態になり、円の1/4しか表示されません。



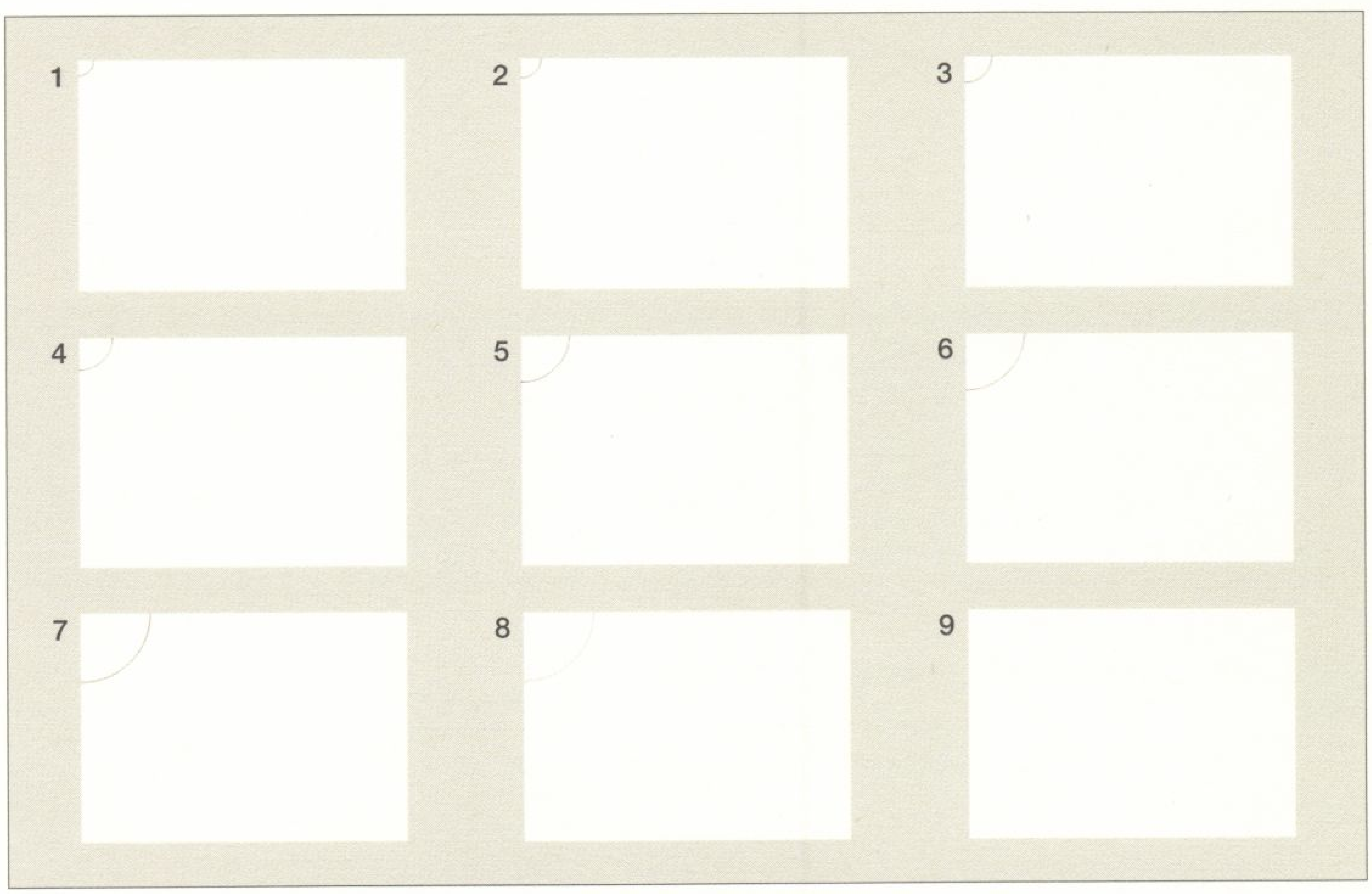
STEP 3 アニメーションが終わったら消す

円のアニメーションが終わったら、画面上から消す処理を追加します。

1 変換したムービークリップ(ripple)の中のタイムラインにアニメーションを付けましょう。
ここでは、円がだんだん大きくなるアニメーションを作ります。大きくなりきった後に、アルファを使って消します。消えたところのフレームにActionScriptでstop( )を設定します。



2 ムービープレビューで確認しましょう。画面では消えて見えなくなっているのですが、ムービークリップは画面に配置されたままです。このままにしておくと、描画にとても負荷が掛かる(※長時間再生すると表示の処理が遅くなる)ので、負荷を減らすために、しっかりと削除することが必要です。



3 removeChildを使って消します。
削除するには、removeChildを使います。

■ removeChild ………………………… P127へ

```
//タイムラインを停止
stop();
//この円を消す
parent.removeChild(this);
```

STEP 4 ENTER_FRAMEを利用する

Flashには、一定間隔で繰り返し同じ処理を行うことができる「ENTER_FRAME」という機能があります。この「ENTER_FRAME」を使ってサンプルのような円を連続して表示する動きを作ります。

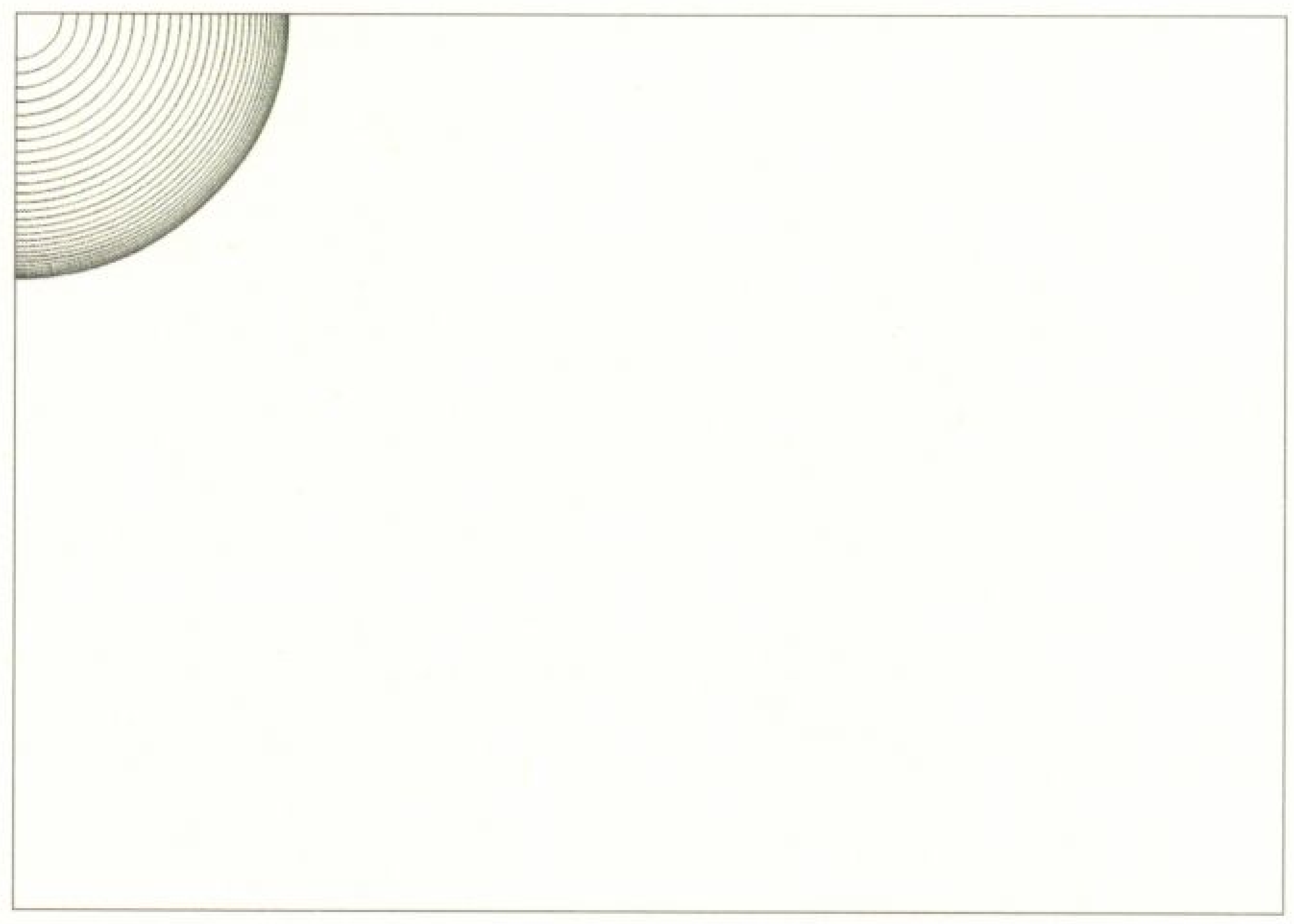
1 メインタイムラインに戻り、function(rippleAnime)の下にaddEventListenerを追加します。

■ ENTER_FRAME ………………………… P121へ

```
function rippleAnime(ev:Event):void{
    trace("rippleAnimeが呼ばれました。");
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, rippleAnime);
```

2 ムービープレビューで確認しましょう。
STEP 3の円の動きを繰り返し、大量に表示しているため円が何重にもなってみえます。



STEP 5 マウスの位置に円を表示する

Flash上のマウスの位置を取得し、円のアニメーションをその位置に表示させる設定をします。

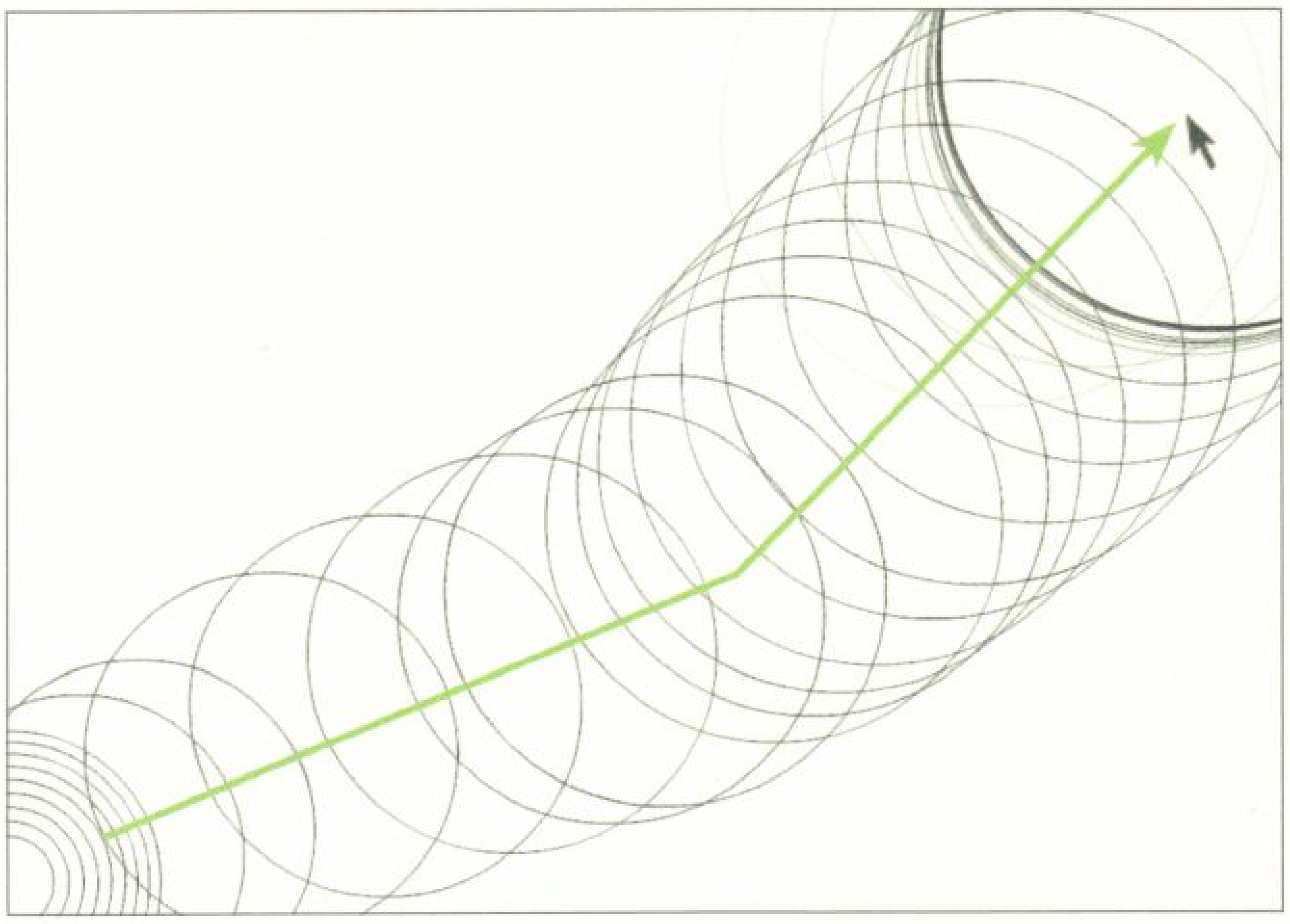
1 Flashムービー上のマウスの位置を取得しましょう。
メインタイムラインのfunction(motion)に右を追加しましょう。

■ mouseX mouseY ……………………… P148へ

```
function rippleAnime(ev:Event):void{
    trace("rippleAnimeが呼ばれました。");
    var ripple:Ripple = new Ripple();
    //マウスの位置にあわせて画面に配置する
    trace("マウスのX位置" + mouseX);
    ripple.x = mouseX;
    trace("マウスのY位置" + mouseY);
    ripple.y = mouseY;
    //画面に配置
    addChild(ripple);
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, rippleAnime);
```

2 プレビューします。こうして、マウスの位置に円のアニメーションが配置されるサンプルの完成です。ここまで作ったサンプルを用意しておきました。[ripple_final.fla]を開いて、動きを確認してみましょう。このサンプルでは、マウスを左下から、右上に移動させています。



STEP 6 円の色を変更する

Flashでは、動的にムービークリップの色を変更することができます。
色の変更には［ColorTransform］を利用します。

1 メインタイムライン上のfunction(motion)の中に右を追加しましょう。
0xffffff（白）にMath.randomを掛け合わせることで、カラフルな色をランダムに作ることができます。

```
//色の変更
//ランダムに色を作る
var color:Number = Math.random() * 0xffffff;
//カラートランスフォームを使う
var transform:ColorTransform = new ColorTransform();
//カラートランスフォームに色を設定
transform.color = color;
//ムービークリップ(ripple)の色の変更
ripple.transform.colorTransform = transform;
```

ActionScriptは最終的に右のようになります。

```
function rippleAnime(ev:Event):void{
    var ripple:Ripple = new Ripple();
    //マウスの位置にあわせて画面に配置する
    ripple.x = mouseX;
    ripple.y = mouseY;
    //色の変更
    //ランダムに色を作る
    var color:Number = Math.random() * 0xffffff;
    //カラートランスフォームを使う
    var transform:ColorTransform = new ColorTransform();
    //カラートランスフォームに色を設定
    transform.color = color;
    //ムービークリップ(ripple)の色の変更
    ripple.transform.colorTransform = transform;
    //画面に配置
    addChild(ripple);
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, rippleAnime);
```

PRACTICE 4

マウスインタラクションを作る②

表現力：★★★☆☆　プログラミング力：★★★☆☆

PRACTICE 3に引き続き、Webサイトに加えるちょっとした演出についてサンプルを使って学んでいきましょう。このサンプルでは、タイムラインを使わずに円を縮小するActionScriptを使ったアニメーションを作ります。また、描画のメモリ使用量の軽減についても、触れていこうと思います。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

- function……………… P086へ
- Timer……………… P140へ
- マウスの位置を取得する……………… P148へ
- addChild……………… P124へ
- addEventListener/removeEventListener……………… P092へ
- Math.random……………… P130へ
- ENTER_FRAME……………… P121へ
- stage.stageWidth / stage.stageHeight……………… P144へ

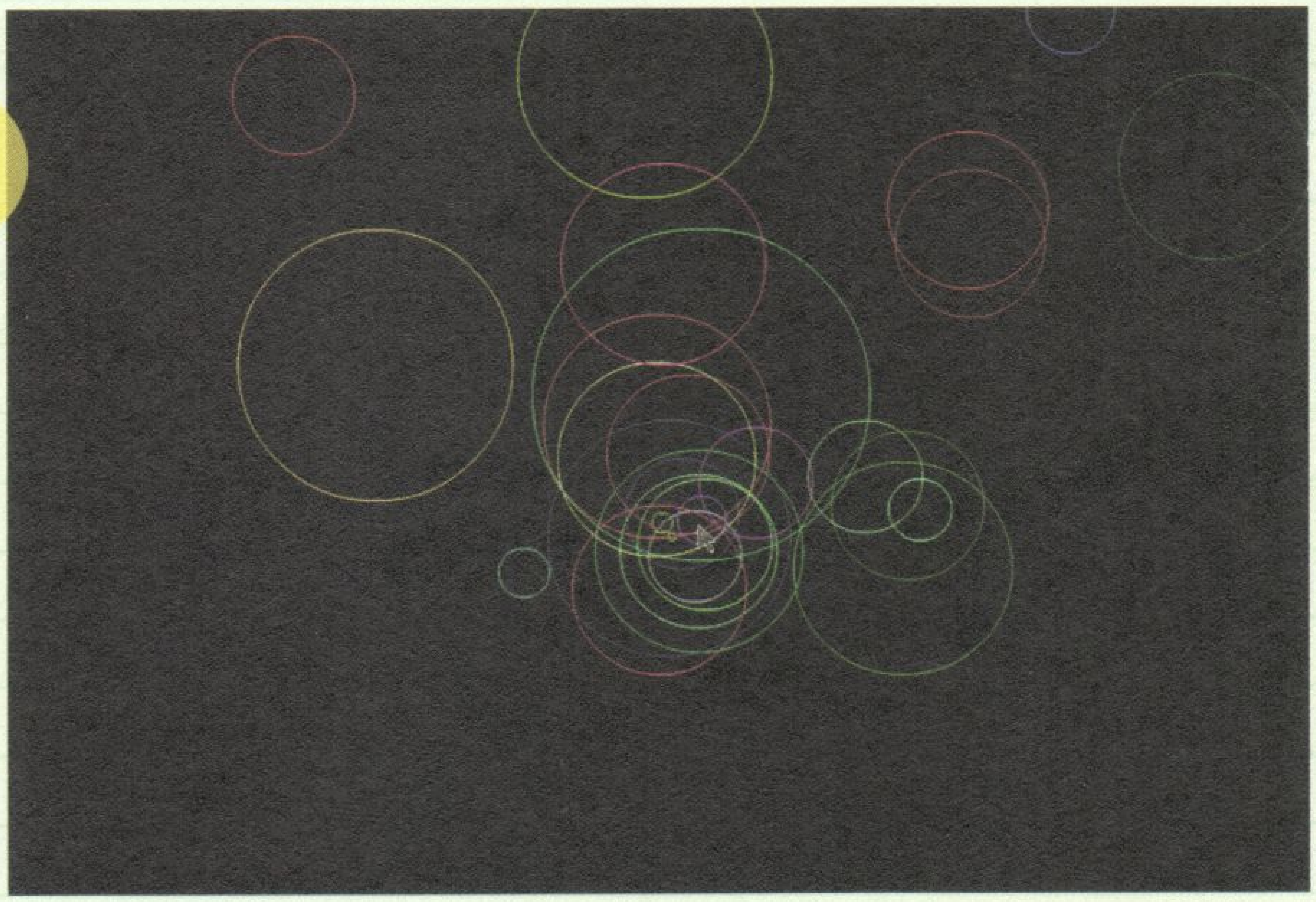
このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

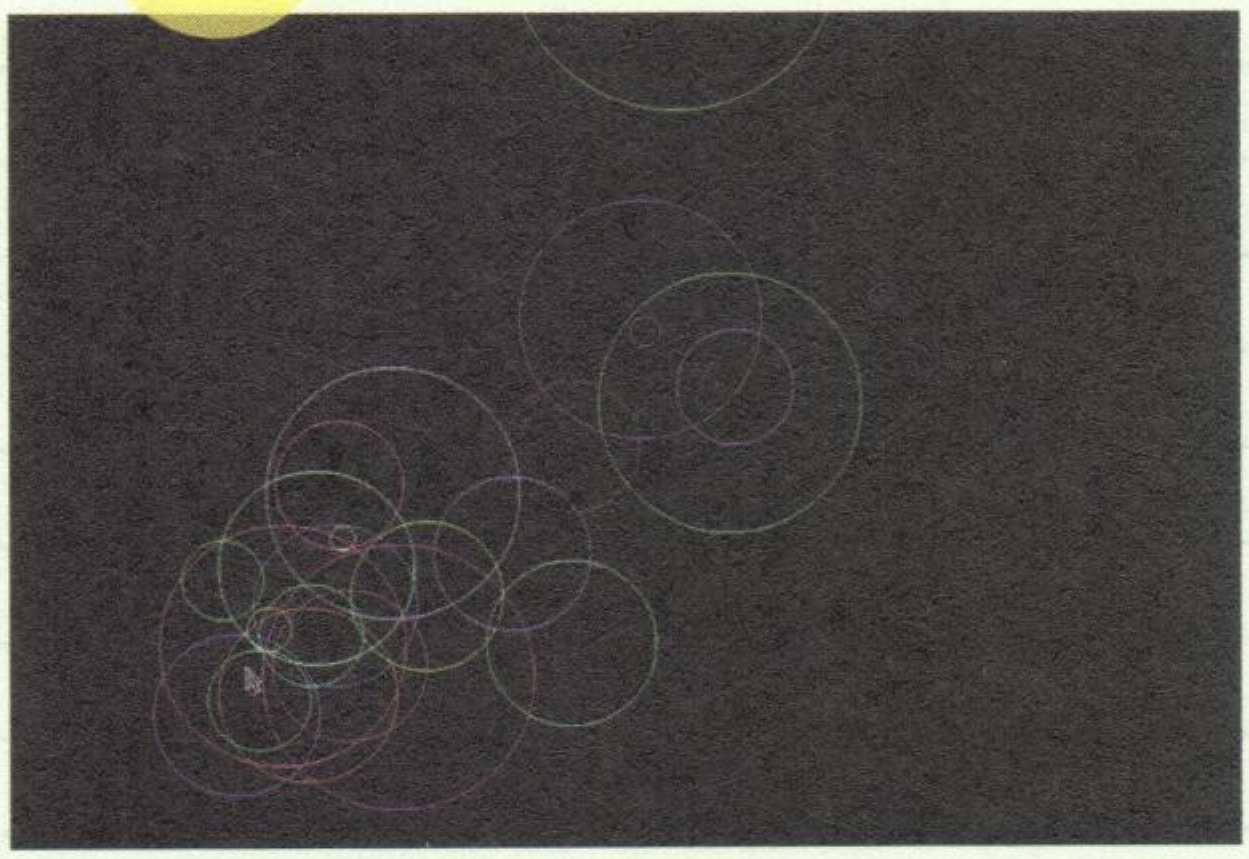
- タイムラインを使わずに（ActionScriptで）行う、円を縮小するアニメーション
- stageWidth、stageHeightを利用して画面の大きさを取得
- Timerを利用したアニメーションの実装
- ENTER_FRAMEとTimerの違いと描画の使用メモリの軽減

サンプルの説明

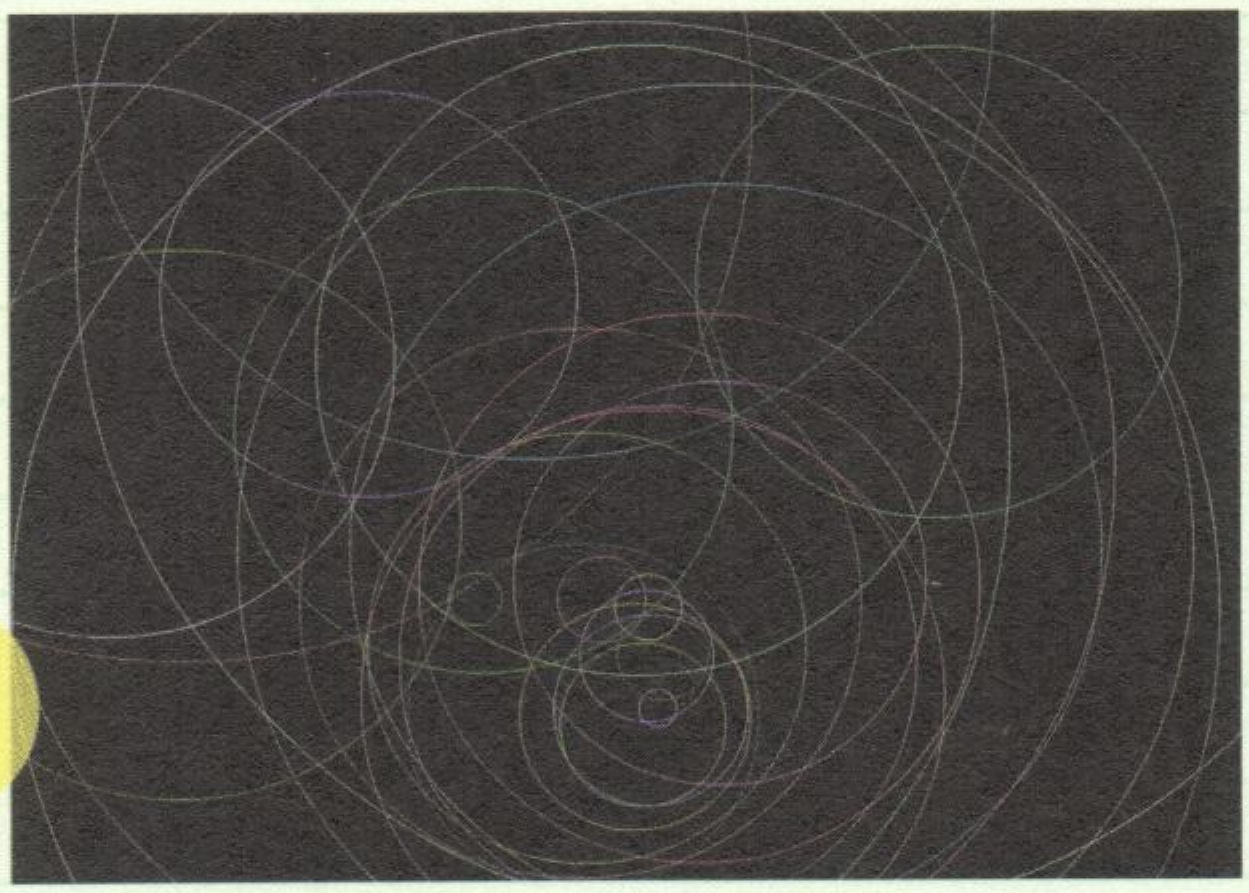
POINT 1



POINT 2



POINT 3



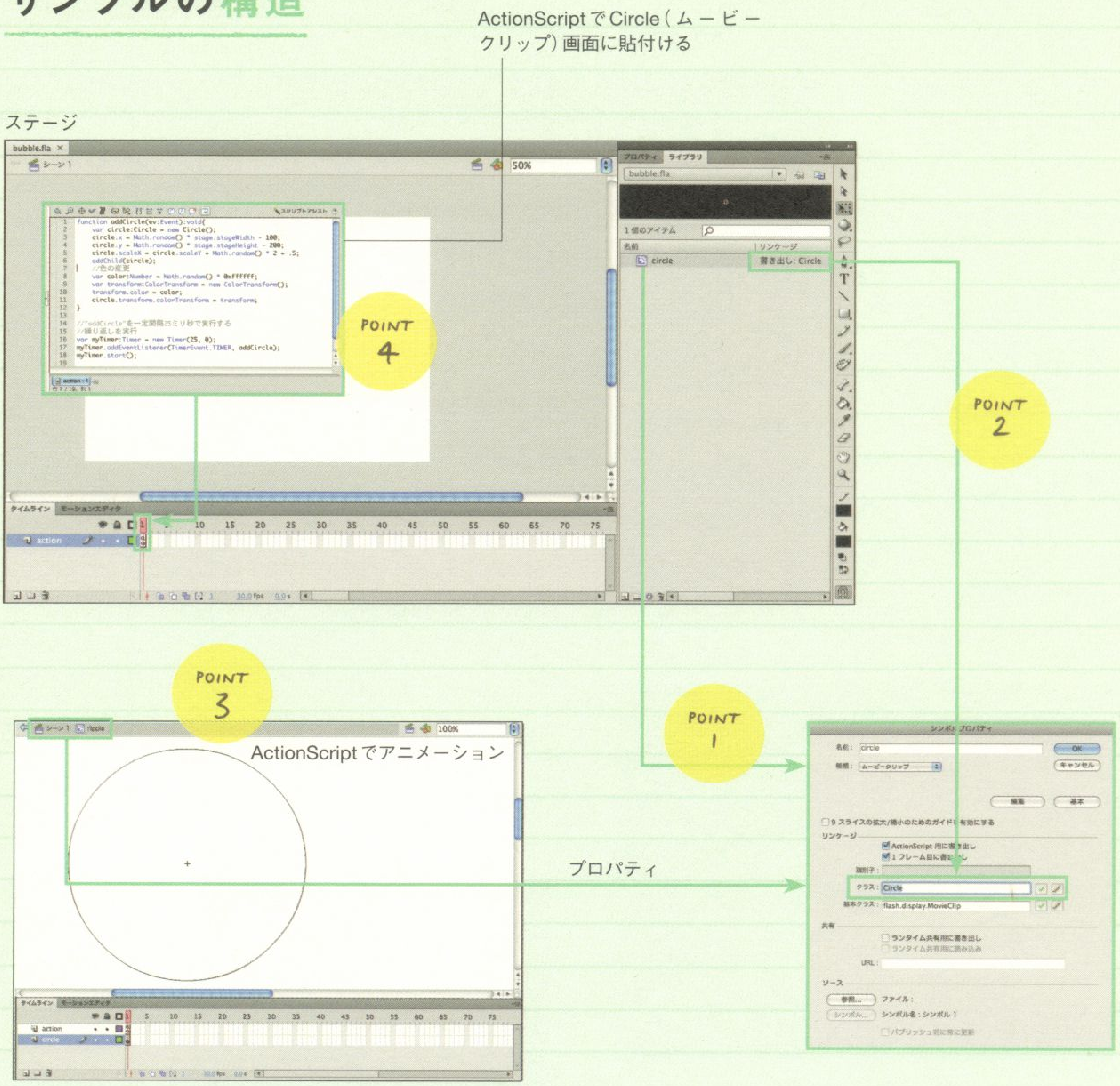
POINT

1 カラフルな円が画面の外側からマウスに向かって集まってきます。

2 円のアニメーションはマウスに向かって縮小し、消えて行きます。

3 上記を繰り返し行います。

サンプルの構造



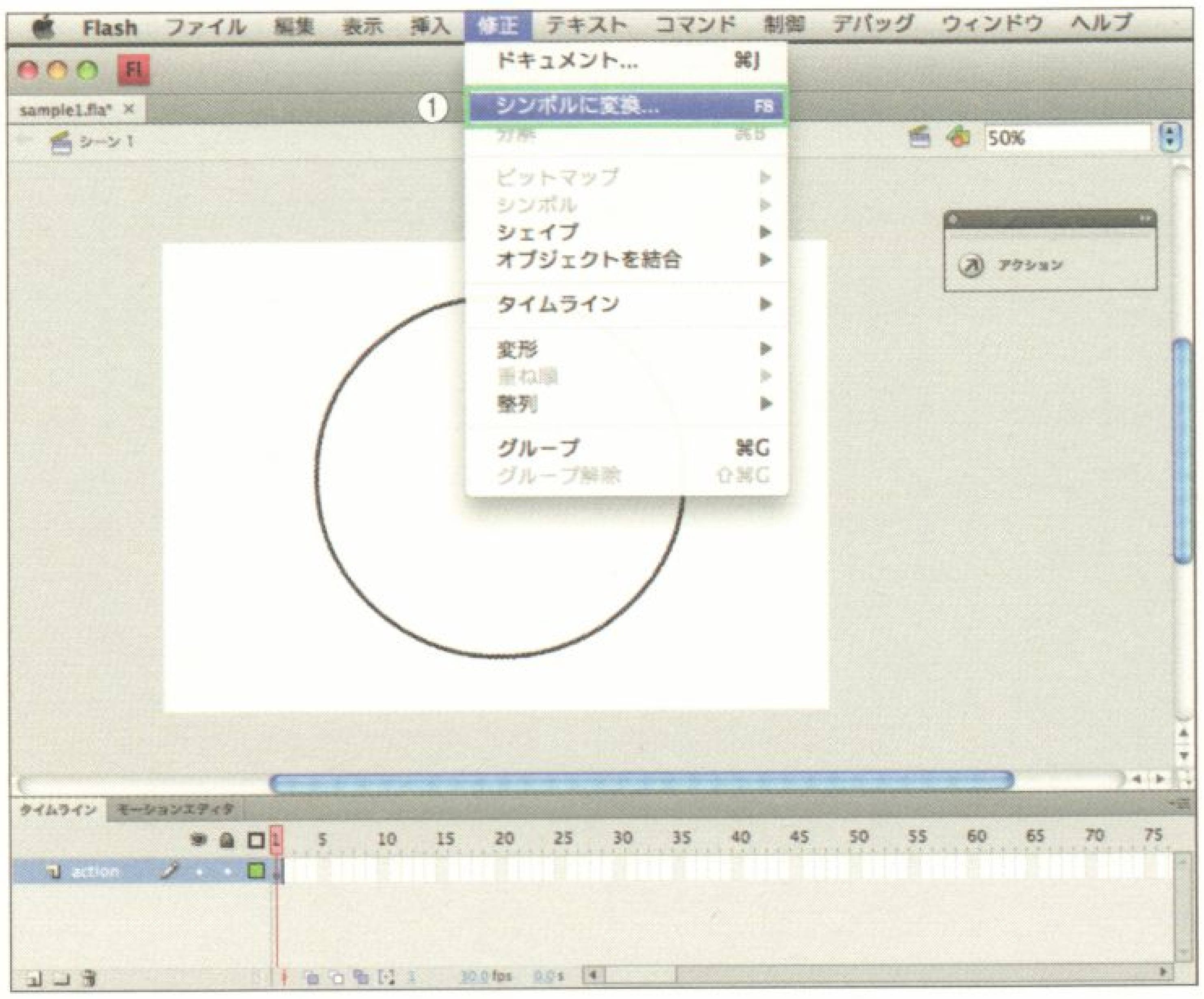
POINT

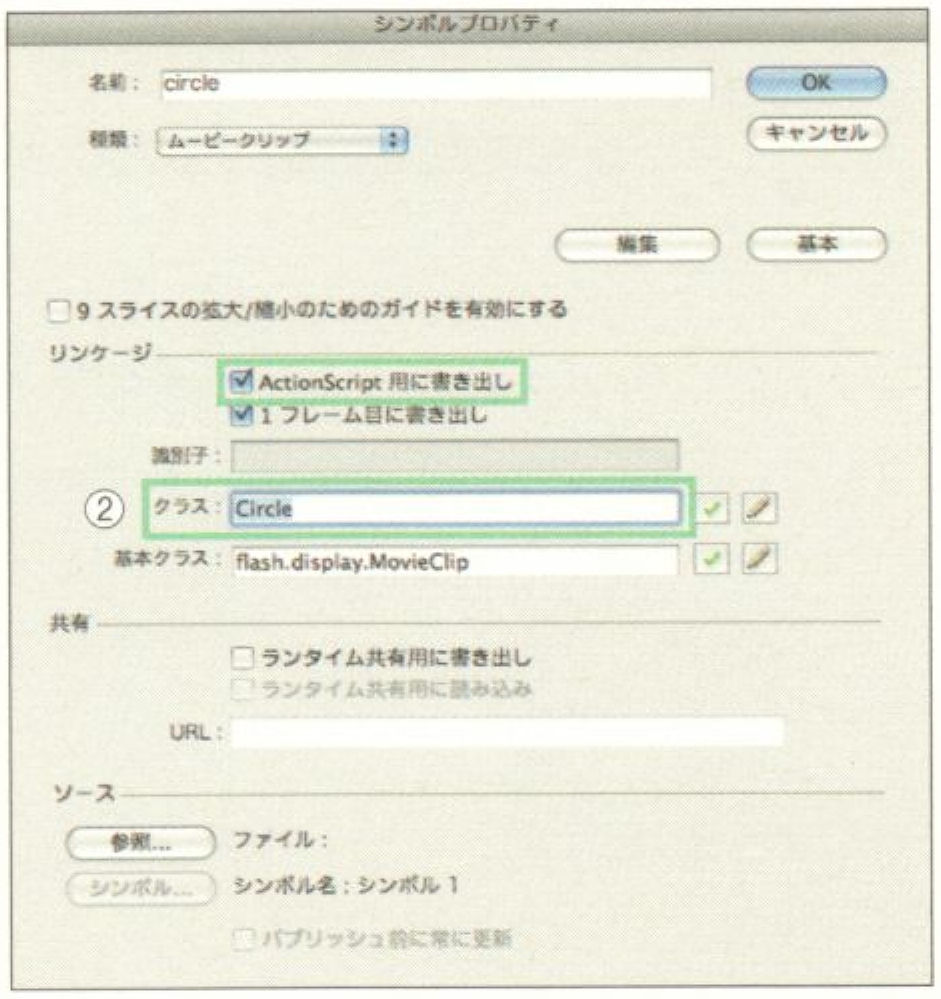
- 1 ランダムに表示される円の元になるムービークリップ(Circle)を用意します。
- 2 ムービークリップ(Circle)をリンケージ書き出しします。
- 3 ムービークリップ(Circle)の中に、円が画面上のマウスの位置へ移動しながら、縮小していくように、ActionScriptを書きます。
- 4 メインタイムラインで、ムービークリップ(Circle)をランダムな大きさと位置に表示させます。ムービークリップ(Circle)を画面に表示（addChild）するだけで、マウスの軌道に合わせて移動します。

STEP 1 ActionScriptで動くアニメーションを作る

ムービークリップの中に、ActionScriptを使って円のアニメーションを作成していきます。

1 アニメーションの元となるムービークリップを作成します。
タイムライン上に円を描き、ムービークリップに変換し(①)、基準点を中心にします。変換するプロパティは下図のように設定します(②)。



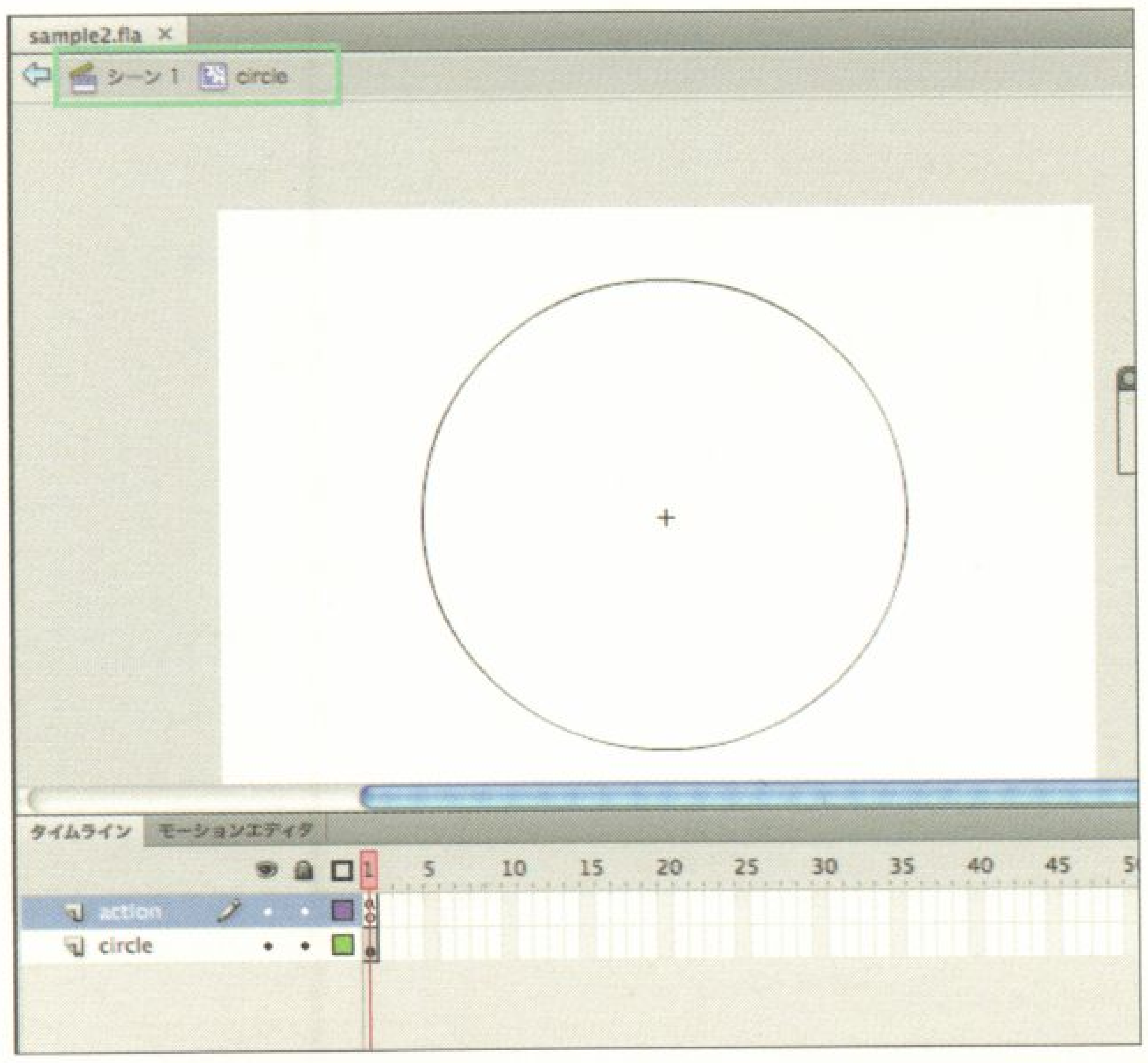


2 ムービークリップをマウスに追従させるように、変換したムービークリップの中のタイムラインの1フレーム目にActionScriptを書いていきます。motionというfunctionを作り、ENTER_FRAMEを利用してこのように動きを付けましょう。

```
//アニメーション用function
function motion(ev:Event):void{
   //ステージ上のマウスのX位置(mouseX)に移動
   x = stage.mouseX;
   //ステージ上のマウスのY位置(mouseY)に移動
   y = stage.mouseY;
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, motion);
```

■ mouseX mouseY ……………………… P148へ
■ ENTER_FRAME ……………………… P121へ



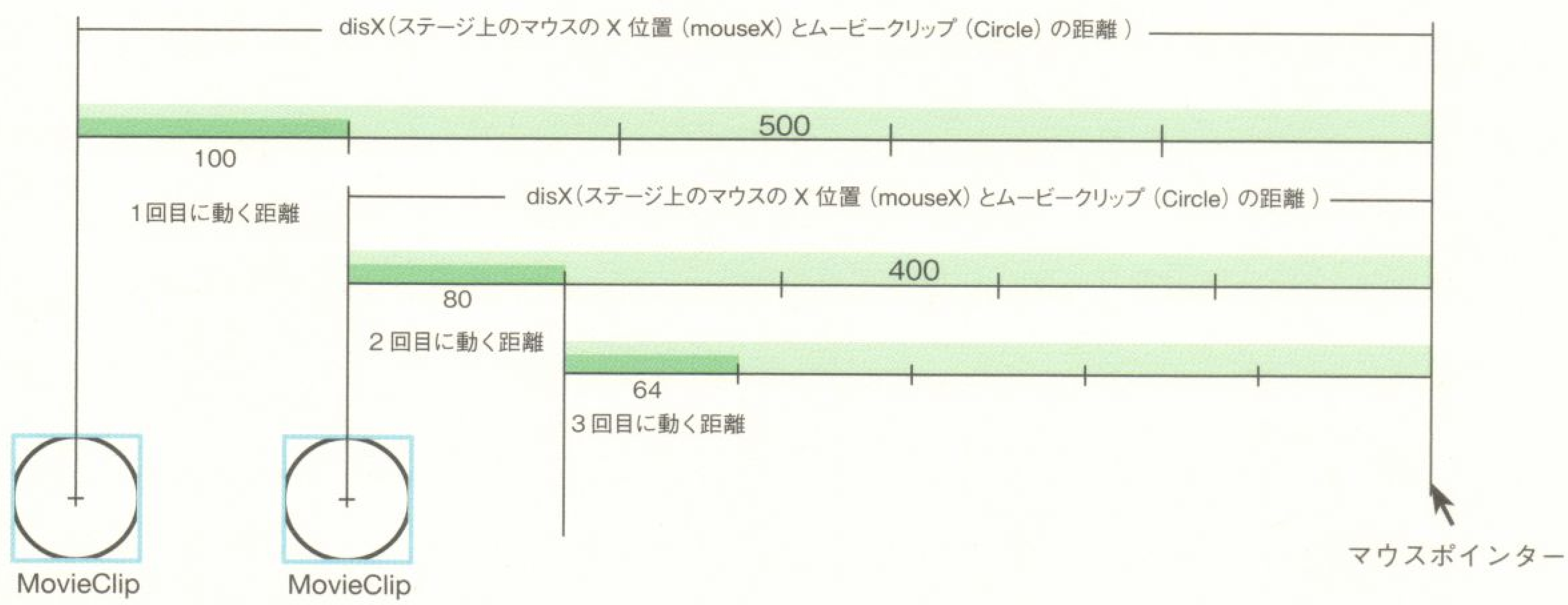
3 これでは、マウスカーソルと同じ動きなので、完成ファイルのように滑らかに円がカーソルに向かってくる動きを付けましょう。

ここで言うスムーズな動きとは、マウスカーソルへ、徐々に近づく動きのことを指します。

徐々に近づくといっても、速度が一定ではなく、はじめは速く、だんだんと減速してマウスに近づきます。

動きはじめが速く、だんだんと減速する動き、イーズアウトの動きをActionScriptで作っていきます。まずは、x軸の動きのみで考え方を見ていきましょう。図解すると以下のようになります。

例：ムービークリップが0pxでマウスポインターが500pxの場合




```
//ステージ上のマウスのX位置(mouseX)との距離
var disX:Number = stage.mouseX - x;
//ステージ上のマウスのX位置(mouseX)との距離の1/5近づく
x = x + disX / 5;
```

はじめにムービークリップ（circle）マウスポインタまでの距離を計り、その距離の1/5を今のムービークリップ（circle）に加算します。

この計算式を繰り返すことで、はじめが速く、だんだん遅くなる動きを作ることができます。

このように、はじめは100。次は80、64、51.2、…… と徐々に移動距離が短くなることで、だんだんと近づく動きを実装できます。

```
//アニメーション用function
function motion(ev:Event):void{
    //ステージ上のマウスのX位置(mouseX)との距離
    var disX:Number = stage.mouseX - x;
    //ステージ上のマウスのX位置(mouseX)との距離の1/5近づく
    x = x + disX / 5;
    //ステージ上のマウスのY位置(mouseY)に移動
    y = stage.mouseY;
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, motion);
```

STEP 2 円のアニメーションが縮小していく動きを付ける

STEP1で作成したムービークリップに対して、マウスカーソルに近づいていくにつれて徐々に小さくなっていく効果を付けます。

1 円をだんだんと小さくしながら、マウスに近づけていきます。先ほどのmotionというfunctionに、右のActionScriptを追加します。

```
//横幅を5%ずつ小さくする
scaleX = scaleX - 0.05;
//縦幅の縮尺を横幅と合わせる
scaleY = scaleX;
```

---

2 実行すると、小さくなりつつ、マウスに近づいたと思いきや急に大きくなるでしょう。ムービークリップを%指定で小さくするscaleX（scaleY）は、ー（負）の値になると左右（上下）に反転して大きくなる特性を持っています。

ムービークリップのscaleXの変化量



---

3 そこで、scaleXが0以下になると、画面から消える処理を追加しましょう。
またここではアニメーションを終了するために、ENTER_FRAMEを消す処理も追加しましょう。
ここまでのActionScriptは右のようになります。

■ removeEventListener ………………… P092へ
■ removeChild ……………………………… P127へ

```
//アニメーション用function
function motion(ev:Event):void{
    //ステージ上のマウスのX位置(mouseX)との距離
    var disX:Number = stage.mouseX - x;
    //ステージ上のマウスのY位置(mouseY)との距離
    var disY:Number = stage.mouseY - y;
    //ステージ上のマウスのX位置(mouseX)との距離の1/5近づく
    x = x + disX / 5;
    y = y + disY / 5;

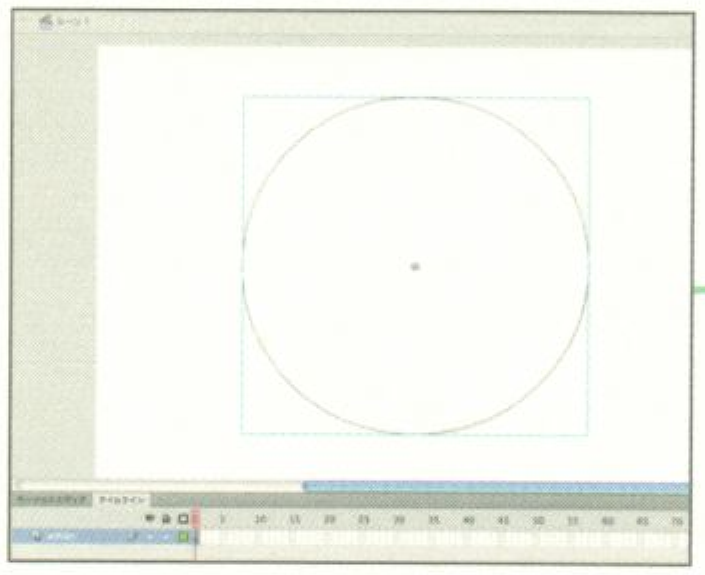
    //横幅を5%ずつ小さくする
    scaleX = scaleX - 0.05;
    //縦幅の縮尺を横幅と合わせる
    scaleY = scaleX;
    //横幅(scaleX)が0以下になると
    if(scaleX < 0){
        //アニメーションを終了
        removeEventListener(Event.ENTER_FRAME, motion);
        //画面からこのムービークリップを削除
        parent.removeChild(this);
    }
}

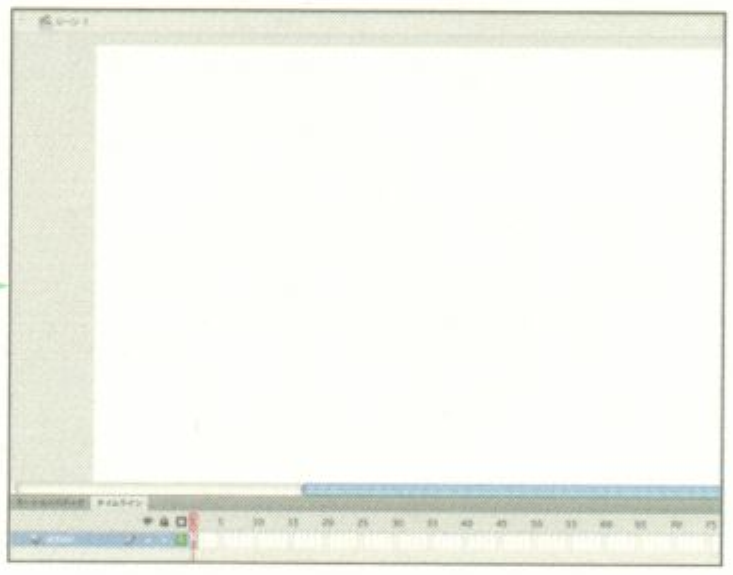
//繰り返しを実行
addEventListener(Event.ENTER_FRAME, motion);
```

STEP 3 Timer関数を使って、画面に配置する

Timer関数を使えば、同じ動作を一定の間隔で連続して実行できます。
STEP 2までで作成した円のアニメーションが連続的に画面に配置されるように設定します。

1 タイムラインにあるムービークリップ(circle)を消して、何も配置していない状態にします。





2 実行すると、ステージに配置されるように、functionにaddChildを追加します。
実行すると1つ画面に配置されて消えていきます。

■ addChild ……………………………… P124へ

```
function addCircle():void{
    var circle:Circle = new Circle();
    addChild(circle);
}

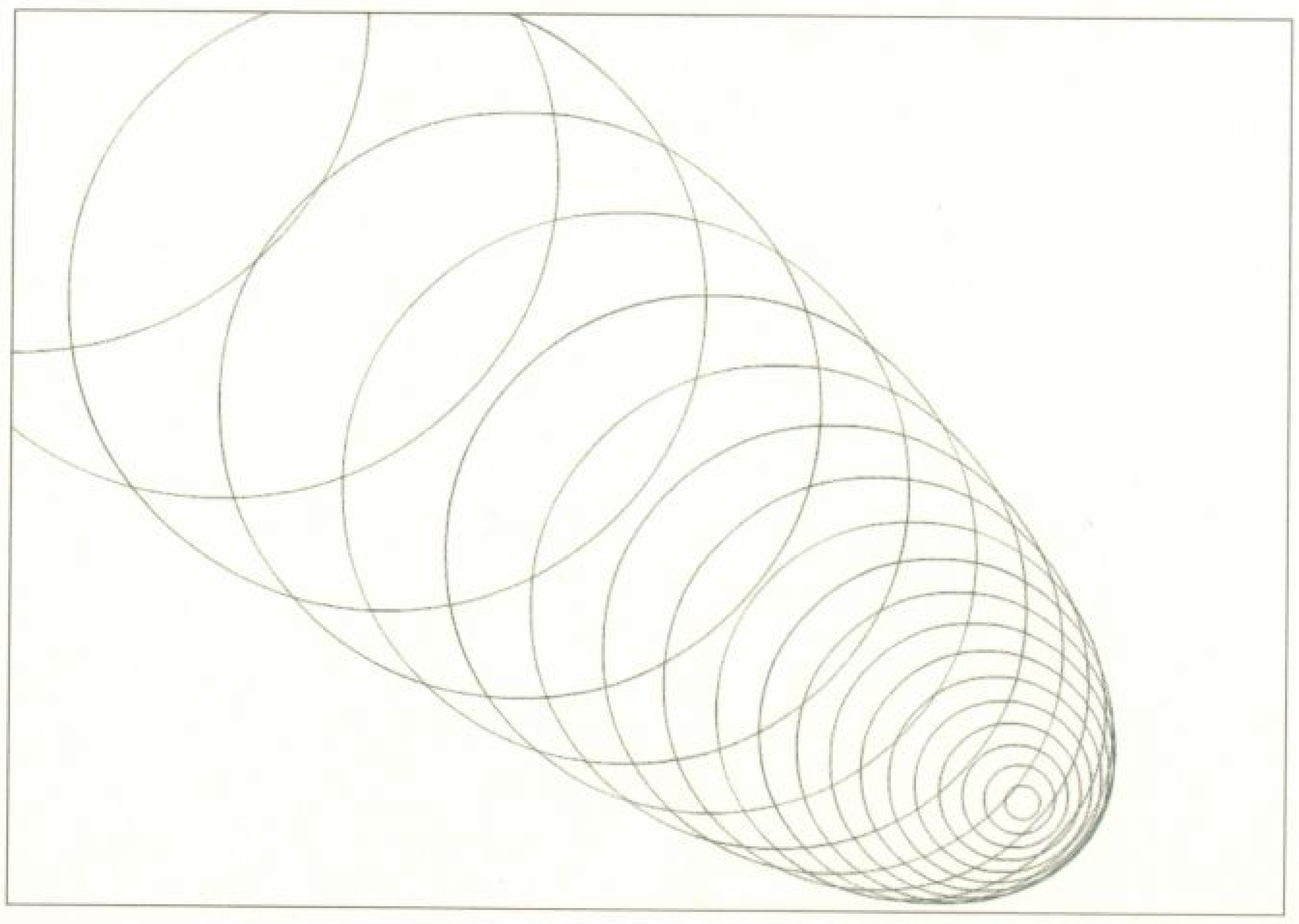
//実行する
addCircle();
```

3 次に、このfunctionをTimer関数で連続的に実行します。functionの( )の中に「ev:Event」を入れるのを忘れないようにしましょう。
実行すると図のようになります。

■ Timer ……………………………… P140へ
■ 引数 ……………………………… P088へ

```
function addCircle(ev:Event):void{
    ------
}

//"addCircle"を一定間隔25ミリ秒で実行する
//繰り返しを実行
var myTimer:Timer = new Timer(25, 0);
myTimer.addEventListener(TimerEvent.TIMER, addCircle);
myTimer.start();
```



STEP 4 画面上にランダムに配置されるようにする

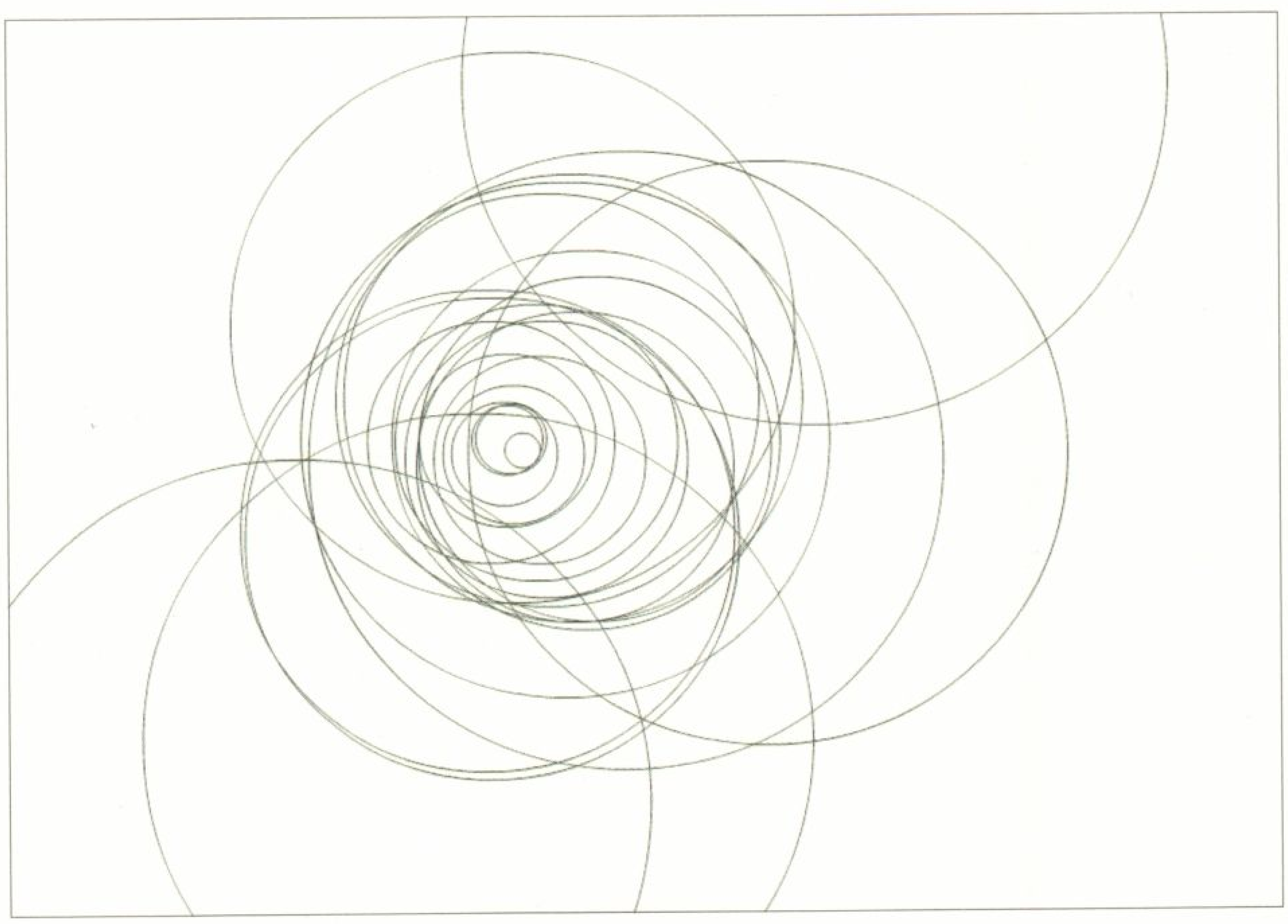
STEP 3までの設定では画面上の同じ位置にしか配置されないので、ランダムな場所に配置されるように ActionScriptを変更しましょう。

1 このままだと、すべて画面の左上に配置されるので、画面上のランダムな場所に配置されるようにしてみましょう。

```
circle.x = Math.random() * stage.stageWidth;
circle.y = Math.random() * stage.stageHeight;
```


- Math.random ······························ P130へ
- stage.stageWidth ························ P144へ
- stage.stageHeight ························ P144へ

2 実行するとこのようになります。



STEP 5 ランダムな色を付け、動きの微調整をする

仕上げとして、円のアニメーションにランダムな色を付け、またサイズもランダムになるように変更します。

1 色の変更には［ColorTransform］を利用します。メインタイムラインのfunction(motion)に右を追加します

```
function addCircle(ev:Event):void{
    var circle:Circle = new Circle();
    circle.x = Math.random() * stage.stageWidth;
    circle.y = Math.random() * stage.stageHeight;
    addChild(circle);
    //色の変更
    var color:Number = Math.random() * 0xffffff;
    var transform:ColorTransform = new ColorTransform();
    transform.color = color;
    circle.transform.colorTransform = transform;
}

//"addCircle"を一定間隔25ミリ秒で実行する
//繰り返しを実行
var myTimer:Timer = new Timer(25, 0);
myTimer.addEventListener(TimerEvent.TIMER, addCircle);
myTimer.start();
```

2 また、サイズもランダムにしてみましょう。同メインタイムラインのfunction(motion)に追加します。

```
//ランダムなサイズにする
circle.scaleX = circle.scaleY = Math.random() * 2 + .5;
```

3 最後に、マウスに近づく速度を少し遅くしてみます。ライブラリからムービークリップ(circle)を選択し、circleのタイムライン上のfunctionを編集しましょう。速度を遅くするために、移動速度を1/5から1/10に変更します。

```
x = x + disX / 10;
y = y + disY / 10;
```

4 最終的にメインタイムラインのスクリプトと、ムービークリップのスクリプトは次のようになります。

メインタイムライン

```
function addCircle(ev:Event):void{
    var circle:Circle = new Circle();
    circle.x = Math.random() * stage.stageWidth;
    circle.y = Math.random() * stage.stageHeight;
    circle.scaleX = circle.scaleY = Math.random() * 2 + .5;
    addChild(circle);
    //色の変更
    var color:Number = Math.random() * 0xffffff;
    var transform:ColorTransform = new ColorTransform();
    transform.color = color;
    circle.transform.colorTransform = transform;
}

//"addCircle"を一定間隔25ミリ秒で実行する
//繰り返しを実行
var myTimer:Timer = new Timer(25, 0);
myTimer.addEventListener(TimerEvent.TIMER, addCircle);
myTimer.start();
```

ムービークリップ

```
//アニメーション用function
function motion(ev:Event):void{
    //ステージ上のマウスのX位置(mouseX)との距離
    var disX:Number = stage.mouseX - x;
    //ステージ上のマウスのY位置(mouseY)との距離
    var disY:Number = stage.mouseY - y;
    //ステージ上のマウスのX位置(mouseX)との距離の1/10近づく
    x = x + disX / 10;
    y = y + disY / 10;

    //横幅を5%ずつ小さくする
    scaleX = scaleX - 0.05;
    //縦幅の縮尺を横幅と合わせる
    scaleY = scaleX;
    //横幅(scaleX)が0以下になると
    if(scaleX < 0){
        //アニメーションを終了
        removeEventListener(Event.ENTER_FRAME, motion);
        //画面からこのムービークリップを削除
        parent.removeChild(this);
    }
}

//繰り返しを実行
addEventListener(Event.ENTER_FRAME, motion);
```

COLUMN

ENTER_FRAMEとTimerの違い

「ENTER_FRAME」は一定間隔でActionScriptを実行するのに最適です。
特にこのサンプルのような、ユーザーのマウスの動きに反応したアニメーションを作る際にとても有効です。
「ENTER_FRAME」は実行スピード(繰り返し実行するスピード)がFPS(フレームレート)に依存するため、このサンプルファイル(FPSが30)ですと、1秒間に30回実行されることになります。
ただし、ここに注意が必要です。パソコンのスペックによるのですが、これだけ多くの回数円を表示するアニメーションを実行するとその分描画に必要なメモリを使用することになります。使用量が一定値を超えると、急に画面の表示が遅くなり、スムーズに表示されなくなってしまうことがあります。
今回のPRACTICE 3で紹介したサンプルも、1秒間に30回実行しなくても、美しい表現に必要な回数はまかなえます。
実行回数を減らすことでこのメモリ使用量を軽減することができるのですが、「ENTER_FRAME」は実行スピード(繰り返し実行するスピード)がFPS(フレームレート)に依存するため、表示する間隔を制御できません。
そこで、このサンプルでは、Timer関数を利用しました。Timer関数は繰り返し実行するタイミングを自由に設定できるので、ActionScriptで描画に使用するメモリを軽減することができます。もし、ENTER_FRAMEを利用したアニメーションで描画が遅くなる現象を確認したならば、このTimer関数を使ってみましょう。

PRACTICE 5

フォトビューワーを作る

表現力：★☆☆☆☆　プログラミング力：★★★★☆

サイトで写真のギャラリーページを見かけることがあると思いますが、それらはFlashで作られていることが多いです。ここではギャラリーページなどでよく使われるようなフォトビューワーを作っていきます。写真が入れ替わるだけなら、アニメーションだけで作ることができるのですが、ActionScriptを使うことで、できるだけ更新がしやすい構造を作成できます。たとえば写真を外部から読み込むことで、たえず最新の写真が表示されるようにしたり、写真の数によってサムネイルの数も動的に変更できたりします。今回は構造の設計方法も重要になるので、その辺りを説明しながら作っていきます。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

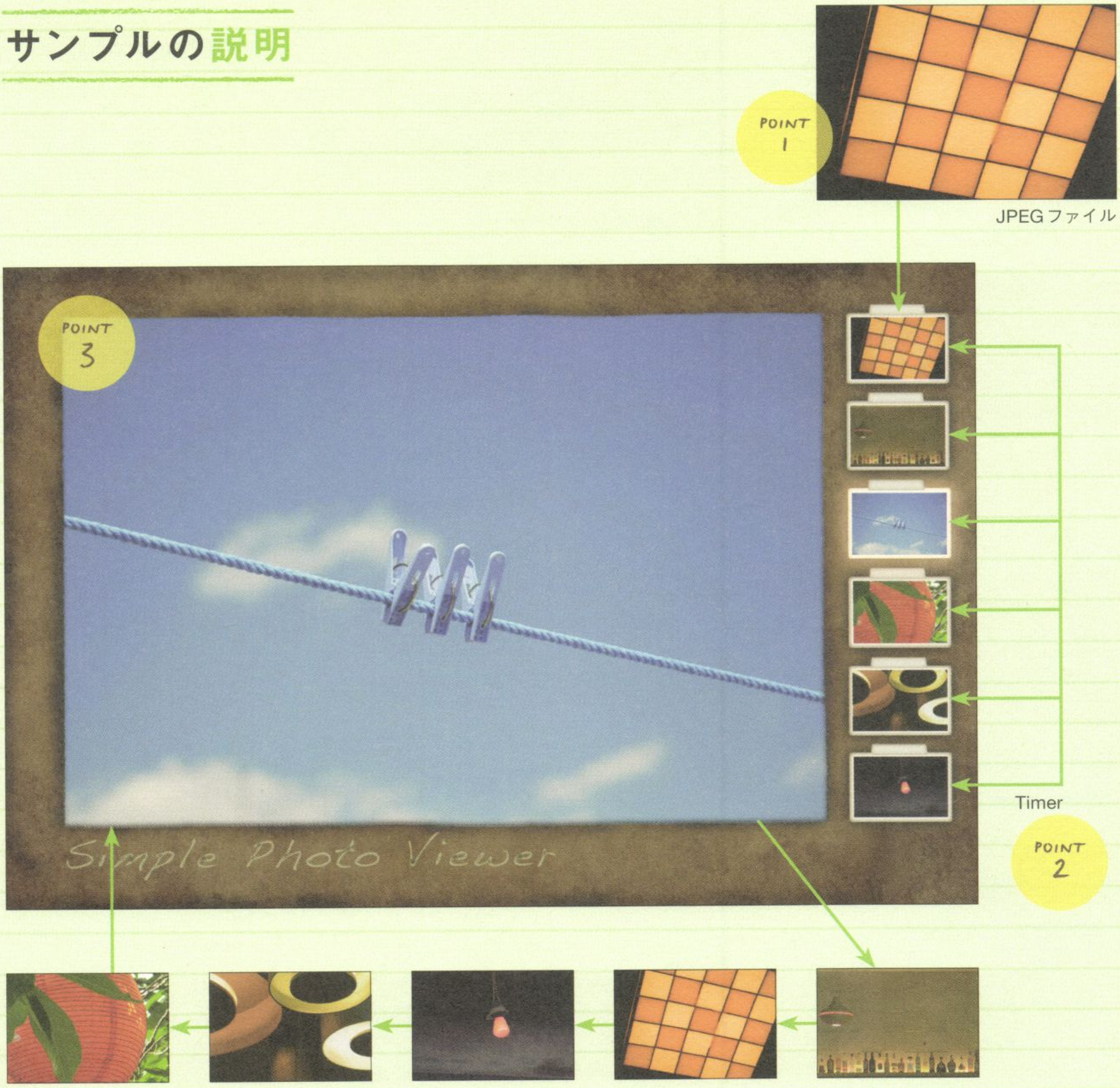
- for ………… P112へ
- 配列 ………… P115へ
- クラス ………… P074へ
- イベントリスナ ………… P090へ
- 外部ファイルを読み込む（Loader） ………… P158へ
- 一定間隔でfunctionを呼び出す（Timer） ………… P140へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- 更新性のある仕組みを組み立てる
- 外部画像からサムネイルを作成する

サンプルの説明



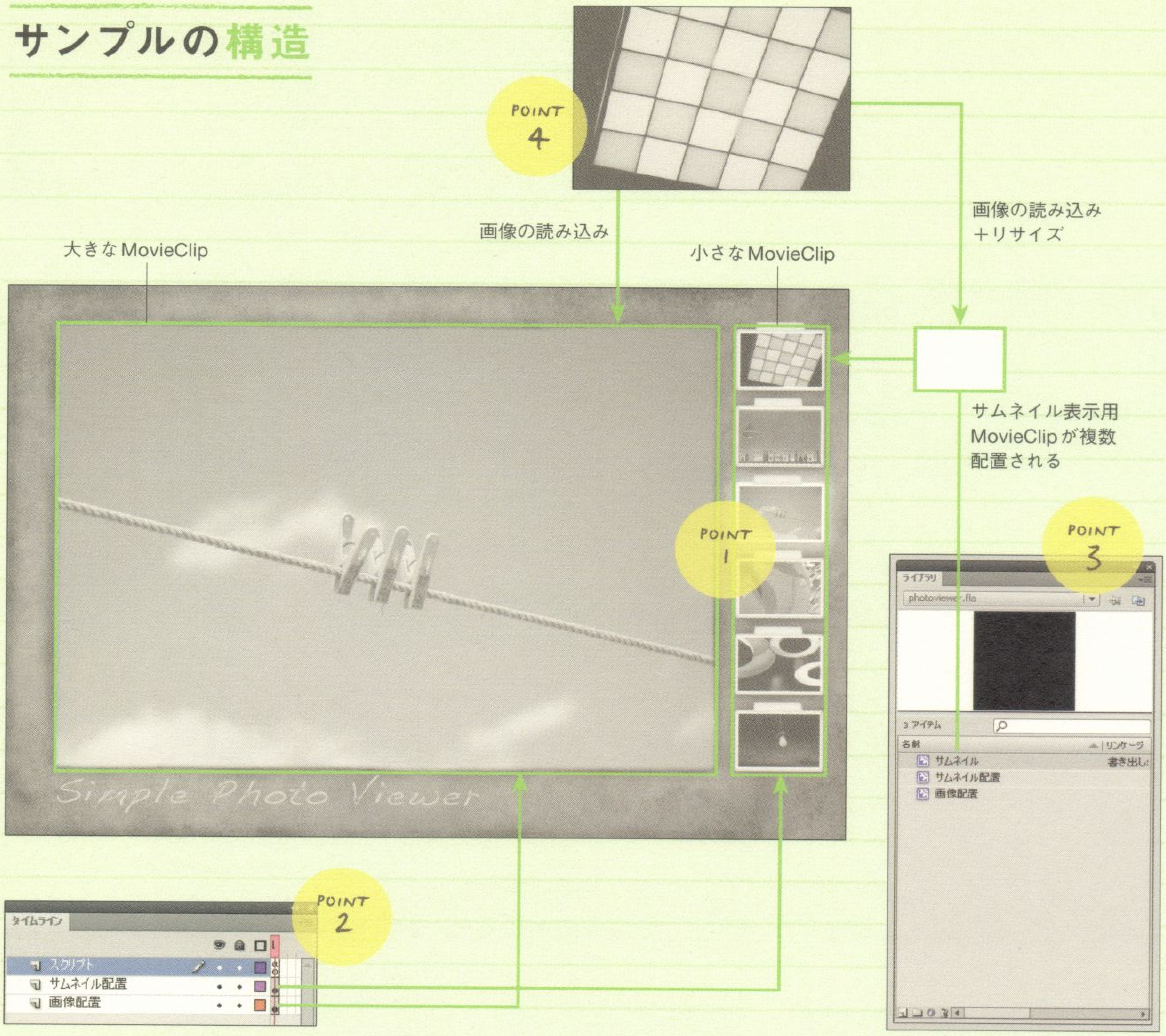
POINT

1 外部ファイルの画像サムネイルが画面の右に表示されます。ここでは外部の画像ファイルを読み込んで、その後リサイズをしています。

2 一定間隔でサムネイルが選択されます。Timerクラスという機能を使い、一定間隔に命令を呼び出します。間隔のタイミングはActionScriptで自由に設定できます。

3 選択されたサムネイルの画像が画面いっぱいに表示されます。自動的に写真が切り替わる仕組みと、サムネイルをクリックして切り替わる仕組みを混合させてみました。

サンプルの構造



POINT

1 メインステージには大きなMovieClipと小さなMovieClipが1つずつ置かれています。大きなMovieClipにはメイン画像を読み込み、小さなMovieClipには複数のサムネイルを読み込みます。

2 タイムラインは1フレームのみです。さきほどのムービークリップが置かれているレイヤーが2つとActionScriptが書かれているレイヤーがあり合計で3つのレイヤーのみです。

3 サムネイルは写真の数により数が増減するので、ActionScriptで配置します。
サムネイル表示用MovieClip自体はライブラリに保存しておきます。

4 サムネイル画像とメイン画像はどちらも同じファイルを読み込んでいます。
サムネイル画像は読み込んだ後にリサイズしています。

STEP 1 画面に要素を配置する

まずはActionScriptでプログラミングしていく前に写真を表示するMovieClipを画面に配置していきましょう。大きく分けて、大きな画像を表示させる場所と、サムネイルを表示させる場所の2つがあります。

1 ステージ上には2つのMovieClipがあります。1つがメイン画像表示用で、もう1つがサムネイル表示用です。サムネイル表示用には複数のサムネイルが動的に配置されます。



サムネイル画像表示用MovieClip

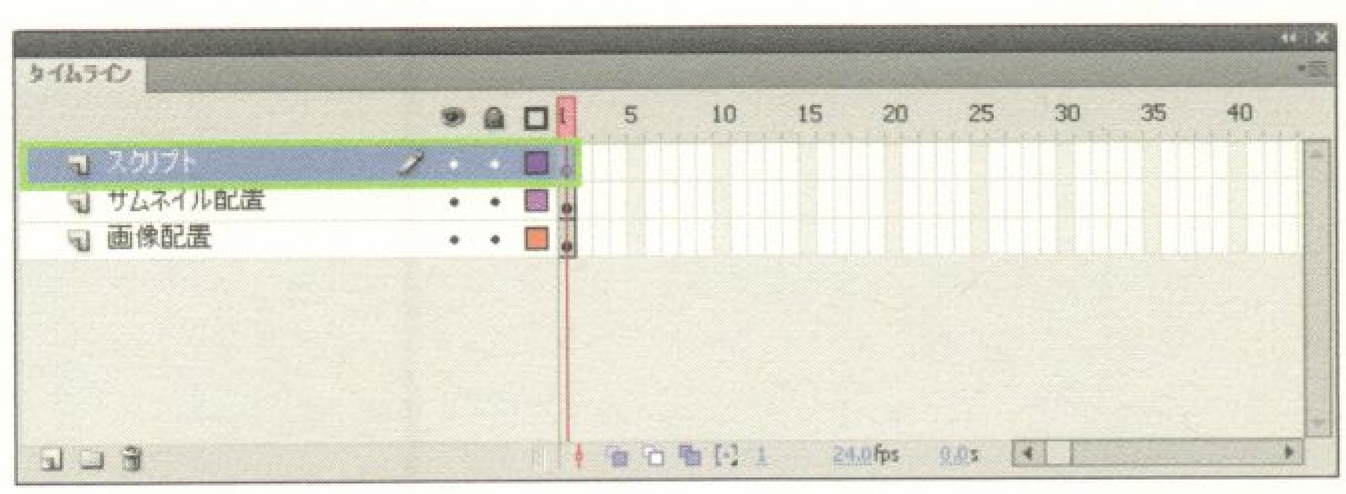
メイン画像表示用MovieClip

それぞれのMovieClipにインスタンス名を付けます。［プロパティ］パネルを開いて、メイン画像用は「mainTarget」、サムネイル用は「thumbnailTarget」とします。





2 もう1つActionScript用レイヤーを追加し、ActionScriptを書いていきましょう。



3 **2**で追加した「スクリプト」レイヤーにここで扱う変数や配列を書きます。

■ 変数 ………………………… P094へ
■ 配列 ………………………… P115へ

```
var photoArr:Array ⇐ 写真のURLを保存する配列
var imageTimer:Timer; ⇐ 写真を自動的に変更するタイマー
var bigImageLoader:Loader; ⇐ 大きな画像を読み込むローダー
var imageNum:Number = 0; ⇐ 現在表示されている画像の番号
```

STEP 2 サムネイルを動的に配置する

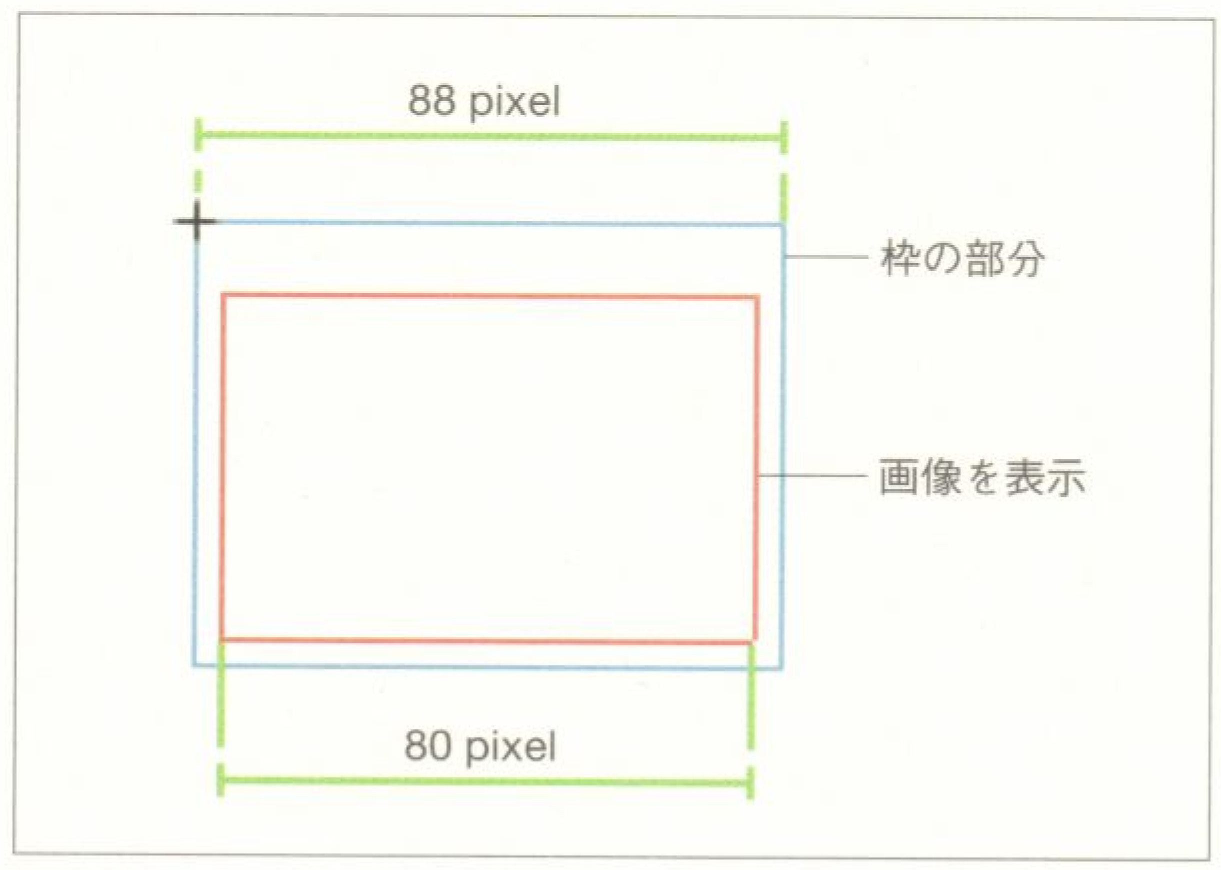
サムネイルを表示するMovieClipをActionScriptで配置します。サムネイル配置で重要なのは、「数が増減するものを動的に配置すること」と「プログラミングで大きな画像を小さくする」ことの2つです。

1 まずは[アクション]パネルを開いて、STEP1で記述した変数の下に実際の配列を作ります。そこに今回読み込みたいファイル名を保存しておきます。ここのファイル数を変更するだけで、サムネイルの数が自動的に変わります。

■ 配列 …………………………………… P115へ

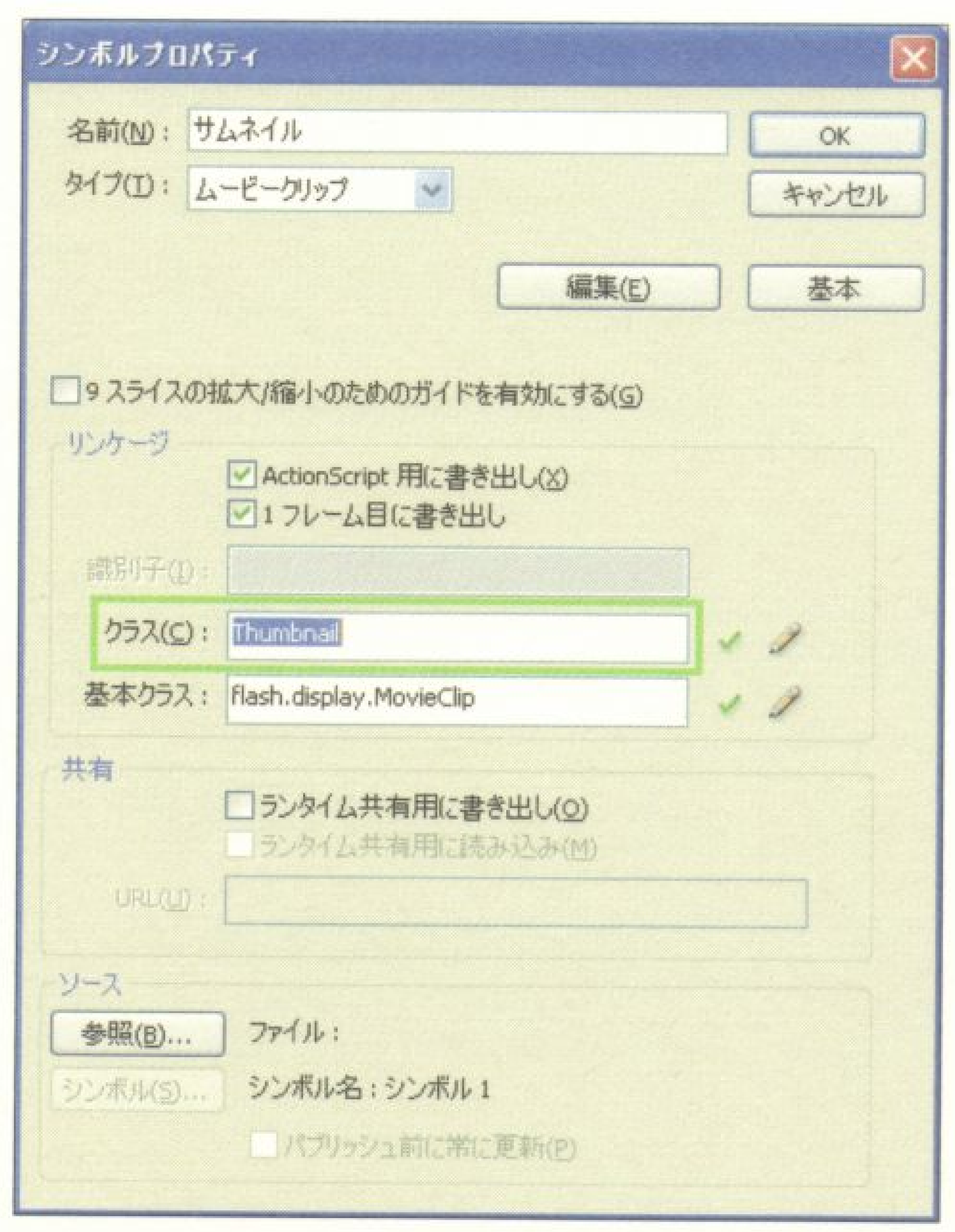
```
// 配列を設定する
photoArr = new Array();
photoArr[0] = "image0.jpg";
photoArr[1] = "image1.jpg";
photoArr[2] = "image2.jpg";
photoArr[3] = "image3.jpg";
```

2 サムネイルを表示させるMovieClipを88×76pixelの四角形で作ります。中身の写真は80×55pixelで表示するため、サムネイルの周囲に枠ができます。
MovieClipの中心点は真ん中にしておきます。



3 「サムネイル」MovieClipのクラス名を「Thumbnail」とします。
今回は実際のThumbnailクラスを生成しません。自分で作成しない場合は、Flashが自動的に作成してくれます。
ここにクラス名を入れるのは、クラス独自の働きをさせるというよりも、ActionScriptで操作するために一意の名前が必要という理由です。

■ クラス …………………………………… P074へ

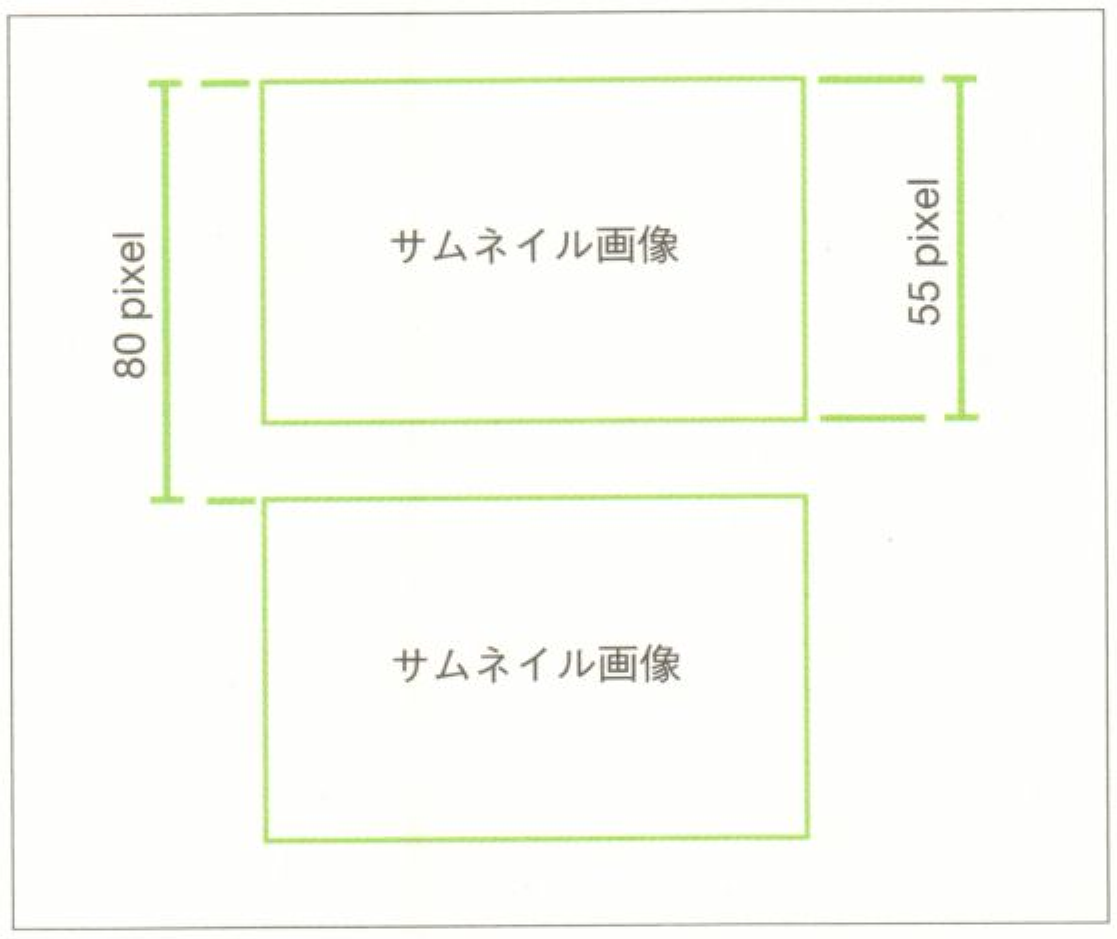


4 forループを使ってサムネイルを配置します。STEP2でThumbnailクラスとサムネイルMovieClipをライブラリで結びつけたため、ActionScriptでこのMovieClipを動的に配置できます。

■配列 …………………… P115へ
■addChild …………………… P124へ

```
// サムネイルを並べる
function setThumbnail() {
  for (var i=0; i<photoArr.length; i++) {
  ⬆ photoArr.lengthで配列の長さを取得できる
    var mc:Thumbnail = new Thumbnail();
  ⬆ このThumbnailクラスがサムネイルMovieClipと結びついている
    mc.x = 4;
    ⬆ 左右にそれぞれ4pixelの幅を空ける
    mc.y = i * 76 + 4;
    ⬆ 80pixelの幅で順番に並べる
    thumbnailTarget.addChild(mc);
    ⬆ thumbnailTargetのMovieClipに配置する
  }
}
```

5 サムネイル画像自体は55pixelですが、80pixelの間隔を空けて配置します。



6 現在のところのコードはこのようになります。

```
// 変数、配列を設定する
var photoArr:Array
var imageTimer:Timer;
var bigImageLoader:Loader;
var imageNum:Number = 0;

// 配列の値を設定する
photoArr = new Array();
photoArr[0] = "image0.jpg";
photoArr[1] = "image1.jpg";
photoArr[2] = "image2.jpg";
photoArr[3] = "image3.jpg";
photoArr[4] = "image4.jpg";

// setThumbnailを実行する
setThumbnail();

// サムネイルを並べる
function setThumbnail() {
  for (var i=0; i<photoArr.length; i++) {
    var mc:Thumbnail = new Thumbnail();
    mc.y = i * 80;
    thumbnailTarget.addChild(mc);
  }
}
```

STEP 3 サムネイルに画像を読み込む

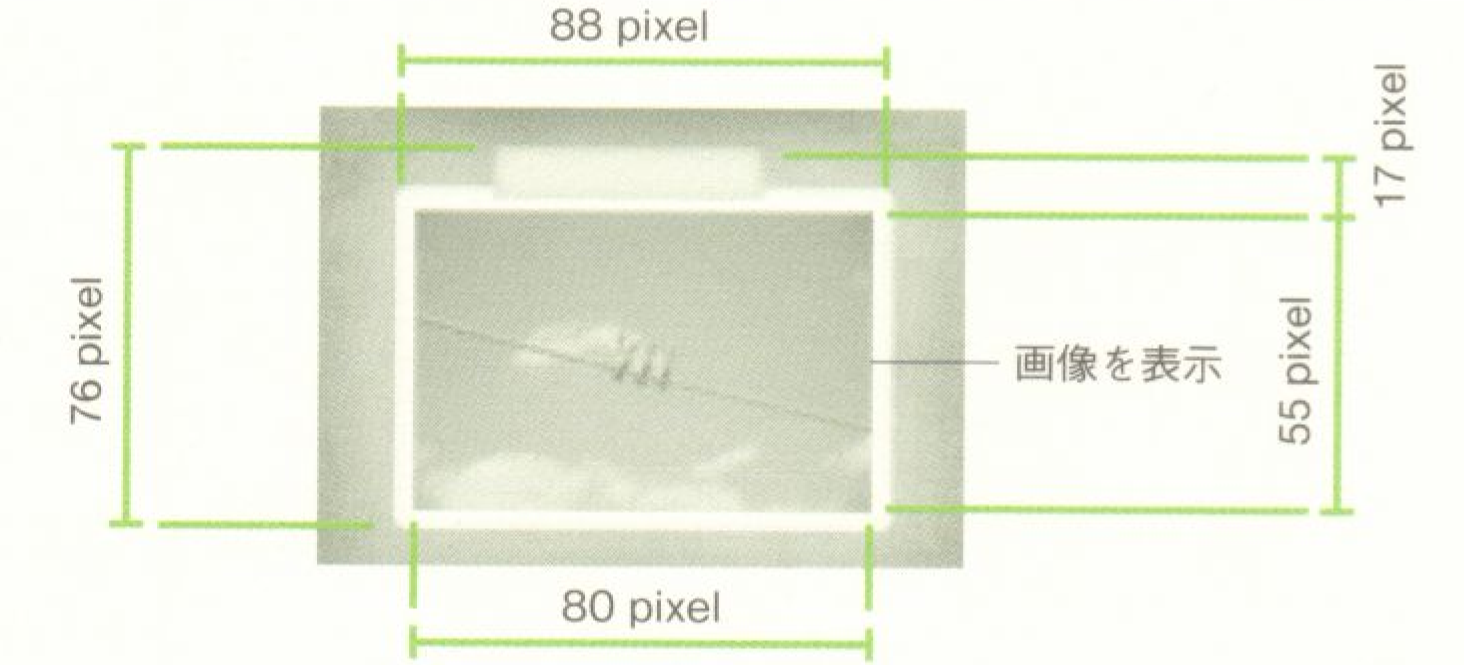
サムネイルMovieClipに画像を読み込みます。読み込む画像はメイン画像と同じものなので、小さくする必要があります。ただし、画像の大きさの変更は画像が読み終わってからでないと変更できません。

1 STEP 2のActionScriptに続けて、画像を読み込むActionScriptを書きます。画像の読み込みにはLoaderクラスを使います。

■Loader ………………………… P158へ

```
// サムネイルを並べる
function setThumbnail() {
    for (var i=0; i<photoArr.length; i++) {
        ....
        // サムネイル画像をロードする
        var loader:Loader = new Loader();
        var url = "images/"+photoArr[i]; ⇐ 写真が置いてあるURLを指定
        var urlRequest = new URLRequest(url);
        loader.load(urlRequest);
        loader.x = 4; ⇐ 画像はサムネイル枠の左端から4pixelずらす
        loader.y = 17; ⇐ 画像はサムネイル枠の上端から17pixelずらす
        mc.addChild(loader);
    }
}
```

2 サムネイルには周囲に枠を付けたいため、左から4pixel離して配置します。サムネイルの枠は88pixelなので、あとで画像を80pixelに変更します。



3 さらに「画像が読み込まれた」ということを知るために、addEventListenerを使って、イベントの通知を待ちます。

■イベントの概念 ………… P077へ

```
function setThumbnail() {
    for (var i=0; i<photoArr.length; i++) {
        ...
        loader.contentLoaderInfo.addEventListener(Event.COMPLETE,
            onThumbnailLoad);
            ⇑ 一番目の引数がイベントの種類で、二番目の引数がイベントが通知され
              た場合に呼ばれる関数
    }
}
```

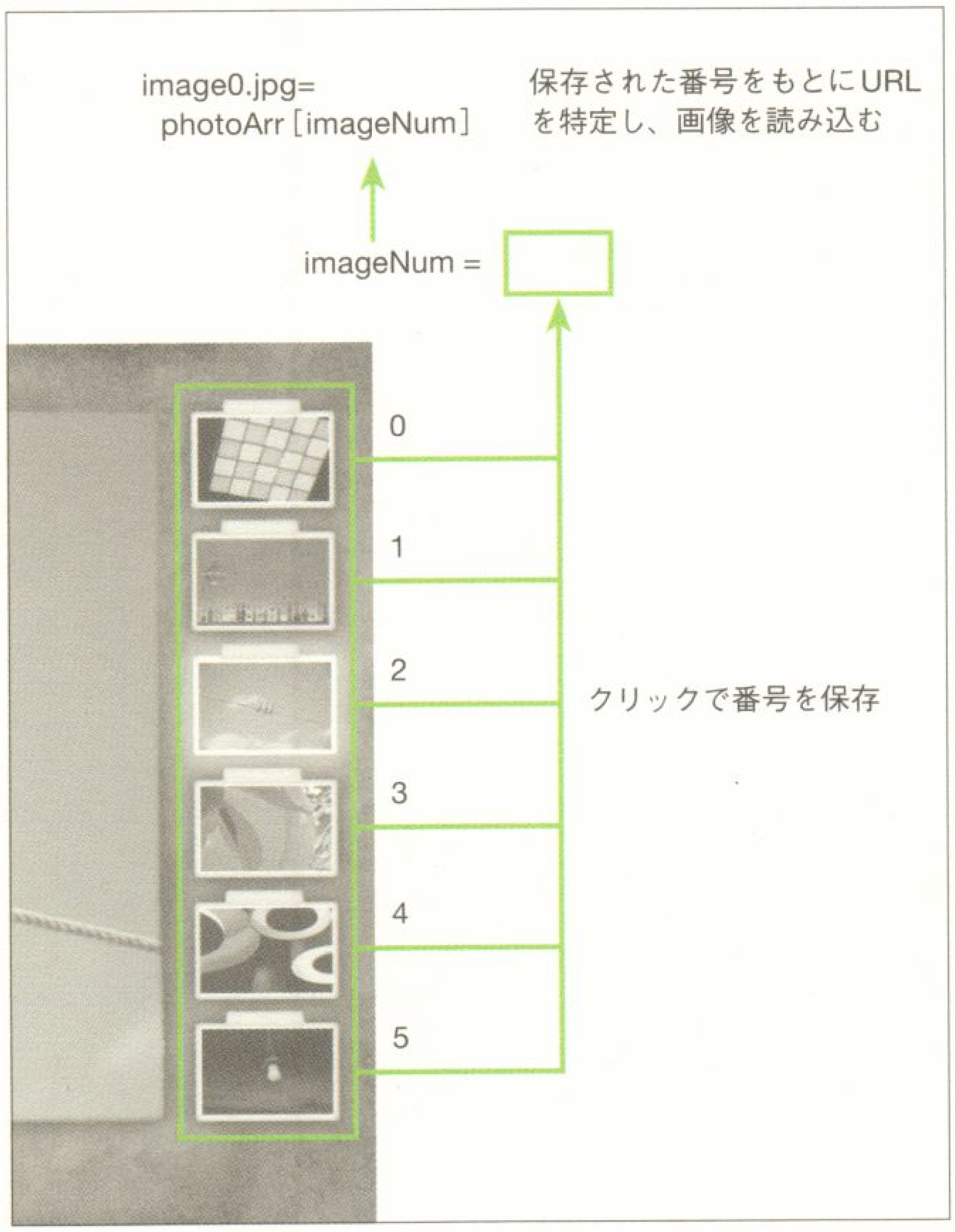
4 画像が読み込まれると指定した関数が呼び出されます。そこで画像の大きさを変更します。

```
// サムネイルがロードされたら実行される
function onThumbnailLoad(eventObj) {
    var mc = eventObj.target.loader;
    ⇑ 画像が置かれている場所は引数の中にある「.target.loader」というところ
      で指定できる
    mc.width = 80;
    mc.height = 55;
    ⇑ 画像の大きさを変更する
}
```

STEP 4 サムネイルをクリックすると大きな画像を読み込む

サムネイルをクリックすると大きな画像を読み込むようにします。一度読み込んだ画像は再度読み込みをしないようにもできますが、少し構造が複雑になるため、今回はクリックされるたびに画像を読み込みます。

1 imageNumという変数を基準として、現在読み込む画像を判断します。



2 Thumbnailクラスをクリックできるようにイベントを登録します。
イベントの登録にはaddEventListenerを使います。

■ イベントリスナ…………………………… P090へ

```
function setThumbnail() {
   for (var i=0; i<photoArr.length; i++) {
      var mc:Thumbnail = new Thumbnail();
      ...
      mc.num = i; ⬅ サムネイルに番号を保存する
      mc.mouseChildren = false; ⬅ サムネイル内の要素押された場合に、内部要素のクリックイベントを無効にする
      mc.addEventListener(MouseEvent.CLICK, onThumbnailClick); ⬅ クリックのイベントを登録する
      mc.buttonMode = true; ⬅ ボタンをロールオーバーした際にマウスを指に変更する
   }
}
```

3 onThumbnailClickでクリックイベントを受け取ります。
ここで現在押されているサムネイルの番号をimageNum変数に保存します。

```
function onThumbnailClick(eventObj) {
    imageNum = eventObj.target.num;
    ⬆ setThumbnailで保存したサムネイルの番号を取得し、現在押されている番号をimageNumに保存する。
    loadImage(); ⬅ 大きな画像をロードするloadImageを呼び出す
}
```

4 loadImageメソッドでは、mainTargetムービークリップに大きな画像を読み込みます。読み込み方法はサムネイルのときと同じです。

```
function loadImage() {
  var url = "images/"+photoArr[imageNum]; ⬅ 保存したimageNumをもとに、photoArrの配列から写真のURLを取得する
  var req = new URLRequest(url);
  bigImageLoader = new Loader(); ⬅ Loaderクラスで画像を読み込む
  bigImageLoader.load(req);
  mainTarget.addChild(bigImageLoader); ⬅ 読み込んだ画像をmainTargetというムービークリップに配置する
}
```

STEP 5 Timerクラスで自動的に写真を切り替える

このフォトビューワーには、サムネイルをクリックしないでも自動的に写真が切り替わる機能があります。ここではその機能を追加します。一定間隔で命令を実行するにはTimerクラスを使用します。

1 タイマーをスタートさせるstartTimerを作成します。ここでは3秒ごとにonTimerCalledを呼び出します。

```
function startTimer() {
  imageTimer = new Timer(3000, 0); ⬅ 3秒(3000ミリ秒)間隔で無限に実行する
  imageTimer.start(); ⬅ タイマーをスタートさせる
  imageTimer.addEventListener(TimerEvent.TIMER, onTimerCalled);
  ⬆ タイマーが実行されるたびにonTimerCalledメソッドが呼び出される
}
```

2 呼び出されるonTimerCalledメソッドを作成します。基本的にはimageNumの番号を繰り上げて、次の画像を読み込みます。

```
function onTimerCalled(eventObj) {
  imageNum++; ⬅ 呼ばれるたびにimageNumの番号を増やしていく
  if (imageNum >= photoArr.length) {
  ⬆ photoArr(写真のURLを保存している配列)の長さより大きくなるとimageNumを0に戻す
    imageNum = 0;
  }
  loadImage(); ⬅ 大きな画像を読み込む
}
```

3 最後にタイムラインの最初にこれらのメソッドを呼び出してあげます。
そうすることでプログラミングが実行します。

```
startTimer(); ← タイマーを起動
setThumbnail(); ← サムネイルを表示
loadImage(); ← 最初の大きな画像を読み込む
```

4 今回のサンプルのコードの全体図です。
初期設定項目を一番上に記述しておき、その次に関数、最後にイベントを受け取る関数を並べています。これがすべて正しい書き方というわけではありませんが、こうして機能ごとにカテゴライズすると、コードの中身が理解しやすいでしょう。

```
//--------------------------------
// 変数、配列の設定
// --------------------------------
var photoArr:Array
var imageTimer:Timer;
var bigImageLoader:Loader;
var imageNum:Number = 0;

//--------------------------------
// 初期化の処理
//--------------------------------
photoArr = new Array();
photoArr[0] = "image0.jpg";
photoArr[1] = "image1.jpg";
photoArr[2] = "image2.jpg";
photoArr[3] = "image3.jpg";
photoArr[4] = "image4.jpg";

//--------------------------------
// 関数の実行
//--------------------------------
startTimer();
setThumbnail();
loadImage();

//--------------------------------
// 関数
//--------------------------------

// サムネイルを並べる
function setThumbnail() {
    // コードは省略
}

// タイマーをスタートする
function startTimer() {
    // コードは省略
}

// 大きな画像を読み込む
function loadImage() {
    // コードは省略
}
```

```
//--------------------------------
// イベントを受け取る関数
//--------------------------------

// サムネイルが押されたら実行される
function onThumbnailClick(eventObj) {
    // コードは省略
}

// サムネイルがロードされたら実行される
function onThumbnailLoad(eventObj) {
    // コードは省略
}

// タイマーが呼ばれるたびに実行される
function onTimerCalled(eventObj) {
    // コードは省略
}

// 大きな画像が読み込まれたら実行される
function onImageLoad(eventObj) {
    // コードは省略
}
```

PRACTICE 6

スクリーンセーバーを作る

表現力：★★★☆☆　プログラミング力：★★★★☆

Flashコンテンツはブラウザで表示するコンテンツに利用されているのはもちろん、その他にもいろいろなシチュエーションで使われています。ここではその中でもスクリーンセーバーをFlashで作っていきます。そして、スクリーンセーバーには起動する度に登録したブログのRSSを取得して表示する機能を付け、毎回起動する度に表示が変化するようにします。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

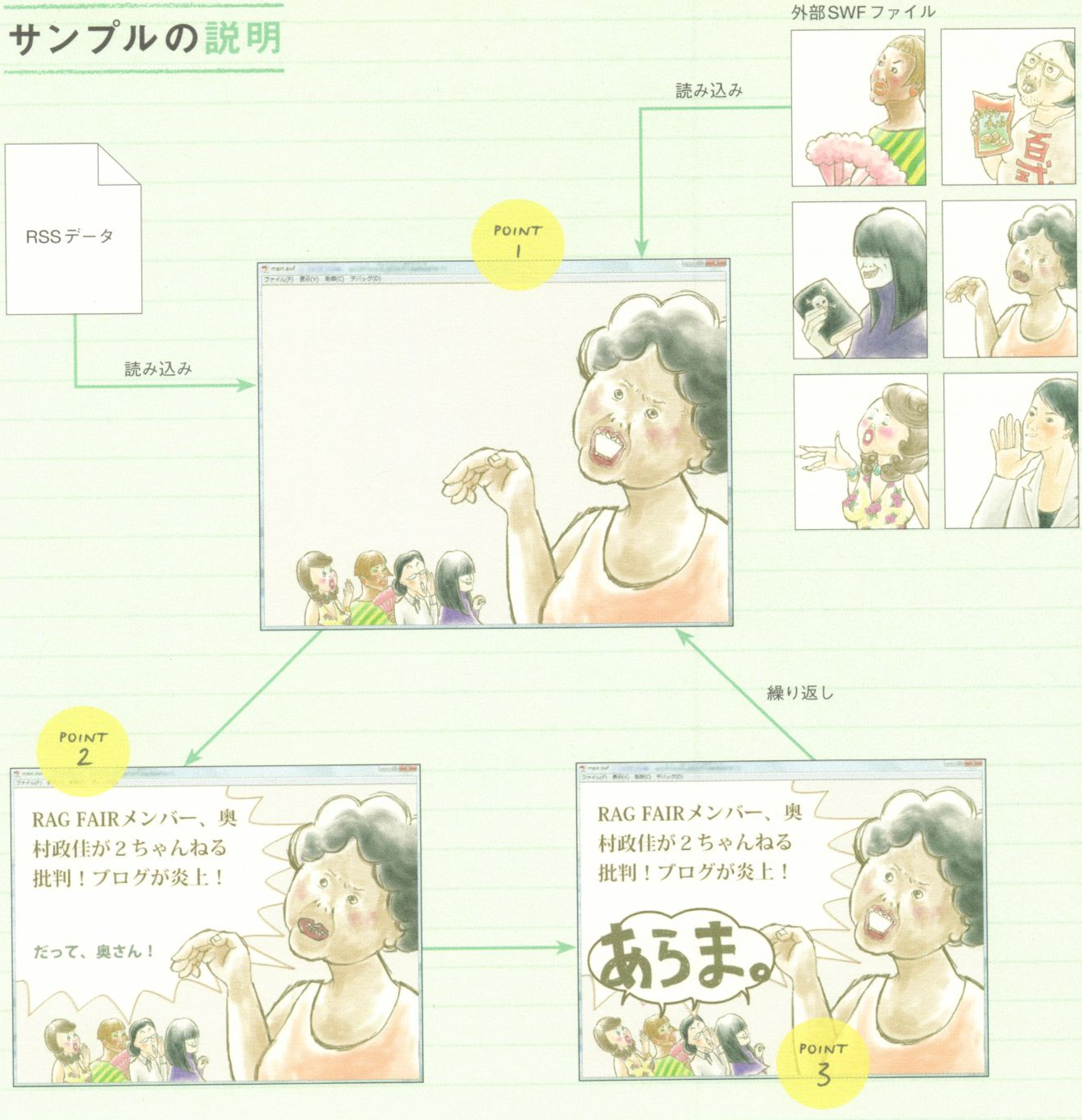
- if ………… P107へ
- for ………… P112へ
- 配列 ………… P115へ
- テキストフィールド ………… P134へ
- クラス ………… P074へ
- 外部ファイルを読み込む（Loader） ………… P158へ
- イベントの取得（addEventListener） ………… P090へ
- 一定間隔でfunctionを呼び出す（Timer） ………… P140へ
- XMLファイルを読み込む（URLLoader） ………… P162へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- RSSデータの読み込み
- 外部SWFファイルの読み込み
- 繰り返し動作するプログラムの考え方
- Flashでスクリーンセーバーを作る方法

サンプルの説明



POINT

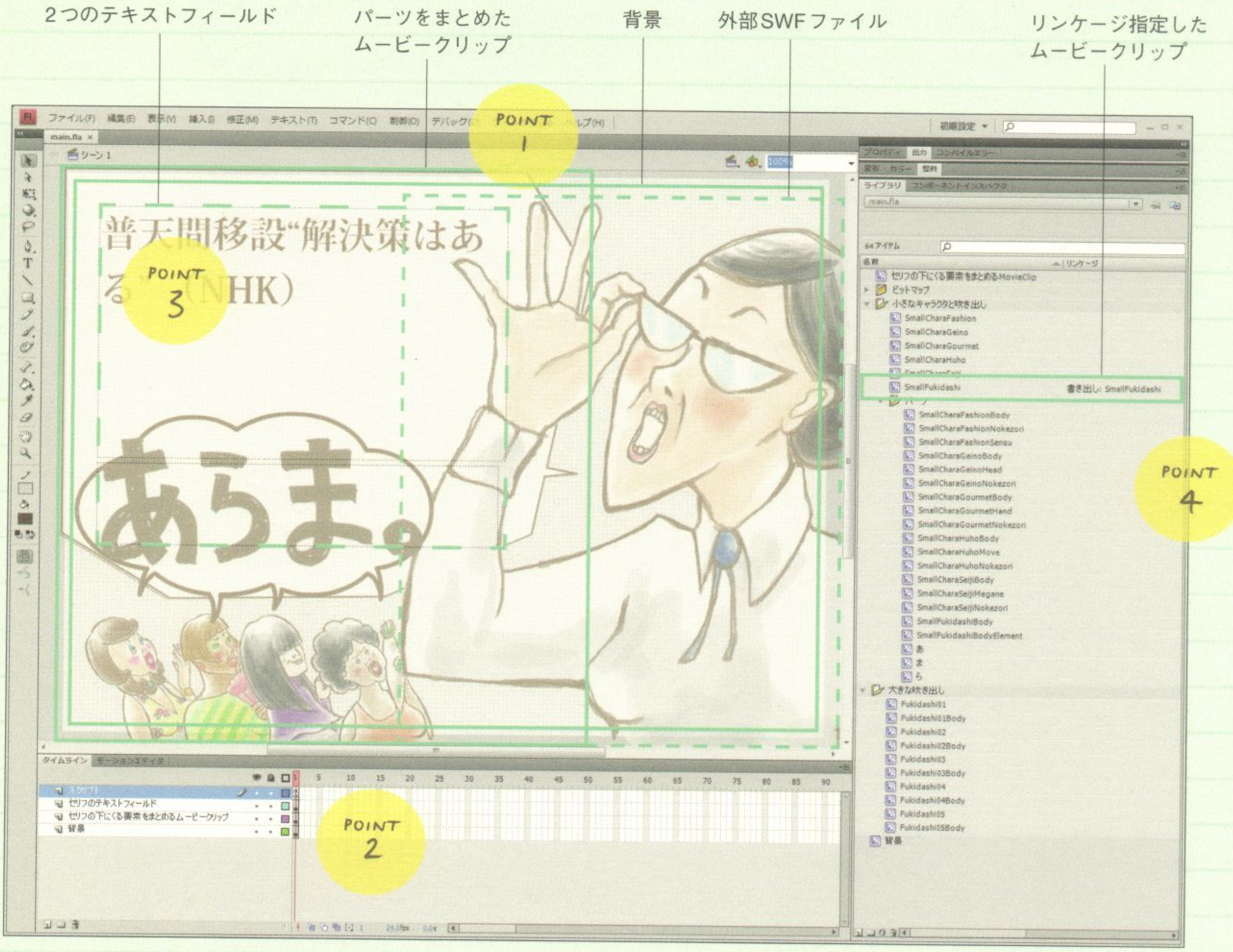
1 複数のRSSをXMLファイルに記載し、動作開始時に読み込んで、RSSに関する情報を読み込みます。そして、そのRSSの中の1つの記事情報をランダムに決め、その記事情報の種類に応じて画面右側にキャラクターを表示させます。このキャラクターは外部SWFファイルとして保持しています。

2 記事情報を吹き出しの中に表示させます。

3 吹き出しの中の記事情報を最後まで表示したら、画面下から小さいキャラクターを表示させます。以後、またランダムに記事情報を取得し、キャラクターを表示させる処理を繰り返します。

PRACTICE 6

サンプルの構造



POINT

1 メインステージには背景となるムービークリップのほかに、吹き出しや小さなキャラクターが入っているムービークリップと、テキストフィールドが2つ配置されています。

2 吹き出しや小さなキャラクターが入っているムービークリップは、テキストフィールドよりも下に重なって表示されるように、レイヤーはテキストフィールドより下にセットします。

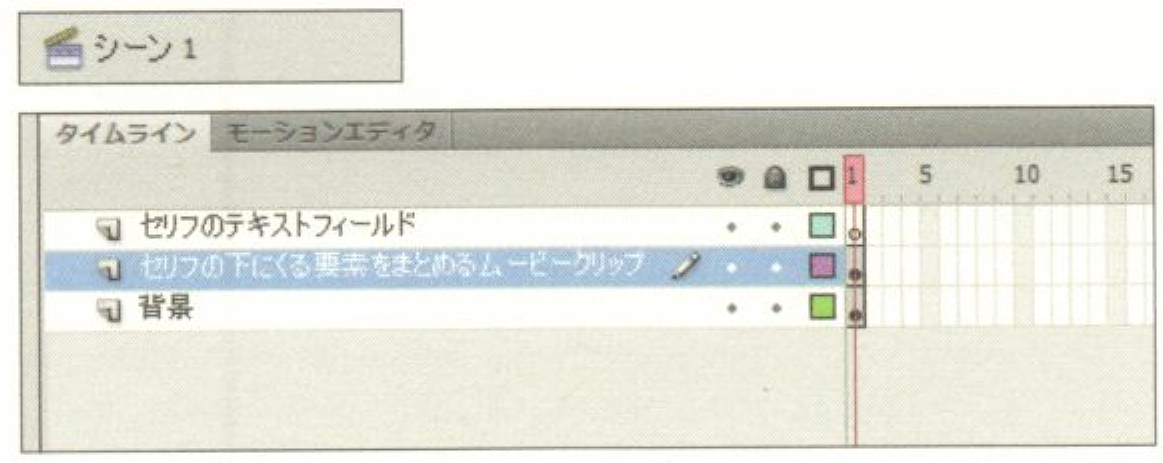
3 2つのテキストフィールドは、いずれも吹き出し内に表示される文字のために配置しています。今回はスクリーンセーバーとしてSWFを利用するので、パブリッシュ後のSWFファイルサイズにはあまりこだわる必要がありません。ですので、これらのテキストフィールドには日本語フォントを埋め込んでいます。

4 ライブラリには配置しているムービークリップや動的に表示するムービークリップ、およびそれらを作るためのパーツムービークリップやビットマップが格納されています。

STEP 1 画面に要素を配置する

まずActionScriptプログラミングをする前に、
あらかじめステージ上に必要なものを配置していきます。

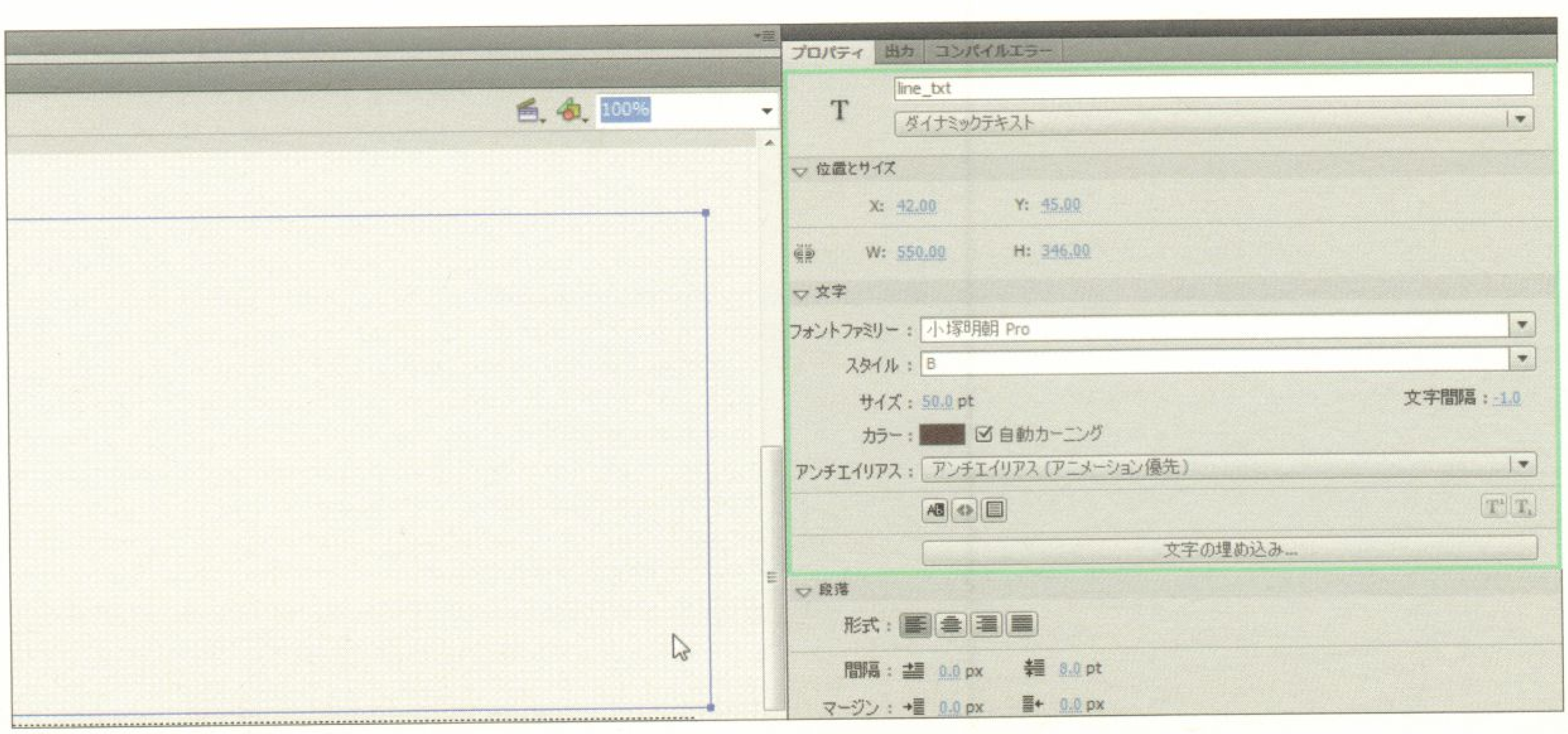
1 まず1つステージ上にレイヤーを用意し、背景のムービークリップシンボルを置きます。そのほか、吹き出しや小さなキャラクターが入っているムービークリップを配置するためのレイヤーと、2つのテキストフィールドを配置するためのレイヤーを作ります。



2 吹き出しや小さなキャラクターを配置するためにムービークリップを作成し、インスタンス名を「container_mc」にします。その中に吹き出しや小さなキャラクターのムービークリップを5つずつ置きます。吹き出しのムービークリップは座標をすべて(0,0)に設定し、小さなキャラクターのムービークリップは表示するタイミングでプログラムから整列させるので、任意の位置に設定します。



3 次に2つテキストフィールド(ダイナミックテキスト)を作成し、画面上のテキストフィールドのインスタンス名を「line_txt」、画面下のテキストフィールドのインスタンス名を「line2_txt」とします。2つのテキストフィールドに日本語フォントを埋め込む設定などを［プロパティ］パネルで行います。今回はRSS経由で表示したい文字が分かるので、基本的にすべての文字セットを埋め込みます。画面上のテキストフィールドには「小塚明朝 Pro」を、画面下のテキストフィールドには「小塚ゴシック Pro」を選択します。



4 ステージ上にもう1つActionScript用のレイヤーを追加し、そこにActionScriptを書きます。はじめに、プログラム全体で使用する変数を書きます。

```
// 設定XMLファイルから取得したデータを格納する配列
var settingDataArray = [];

// ランダムに決める表示中のRSSの配列番号を保存する
var selectedRssNum = 0;

// RSSの記事情報の中で表示中のものの配列番号を保存する
var selectedItemNum = 0;
// ENTER_FRAMEが実行された数をカウントするための変数
var enterFrameCount = 0;

// 小さいキャラクターのインスタンス名すべてを格納する配列
var smallCharaMovieClipArray = ["fashion_mc", "geino_mc", "gourmet_mc", "huho_mc", "seiji_mc"];
```

5 プログラムの動作を開始させるときに、まず最初に行う処理のことを「初期化(initialize:イニシャライズ)」処理と呼びます。このサンプルでは、container_mcの中に配置した吹き出しや小さなキャラクターのムービークリップを非表示にする処理と、テキストフィールドの中身を空にする処理を行っています。そして、初期化処理が終わったら、最初に動作させる関数を実行して、次に動作をつなげます。

```
// 初期化関数を実行
init();


/**
 * 初期化
 */
function init()
{
  // テキストフィールドの中身を空にしておく
  line_txt.text = "";
  line2_txt.text = "";

  // 小さいキャラクターのムービークリップをすべて非表示にする
  for(var i = 0; i < smallCharaMovieClipArray.length; i++) {
    var mcName = smallCharaMovieClipArray[i];
    var mc = container_mc.getChildByName(mcName);
    mc.visible = false;
    mc.stop();
  }

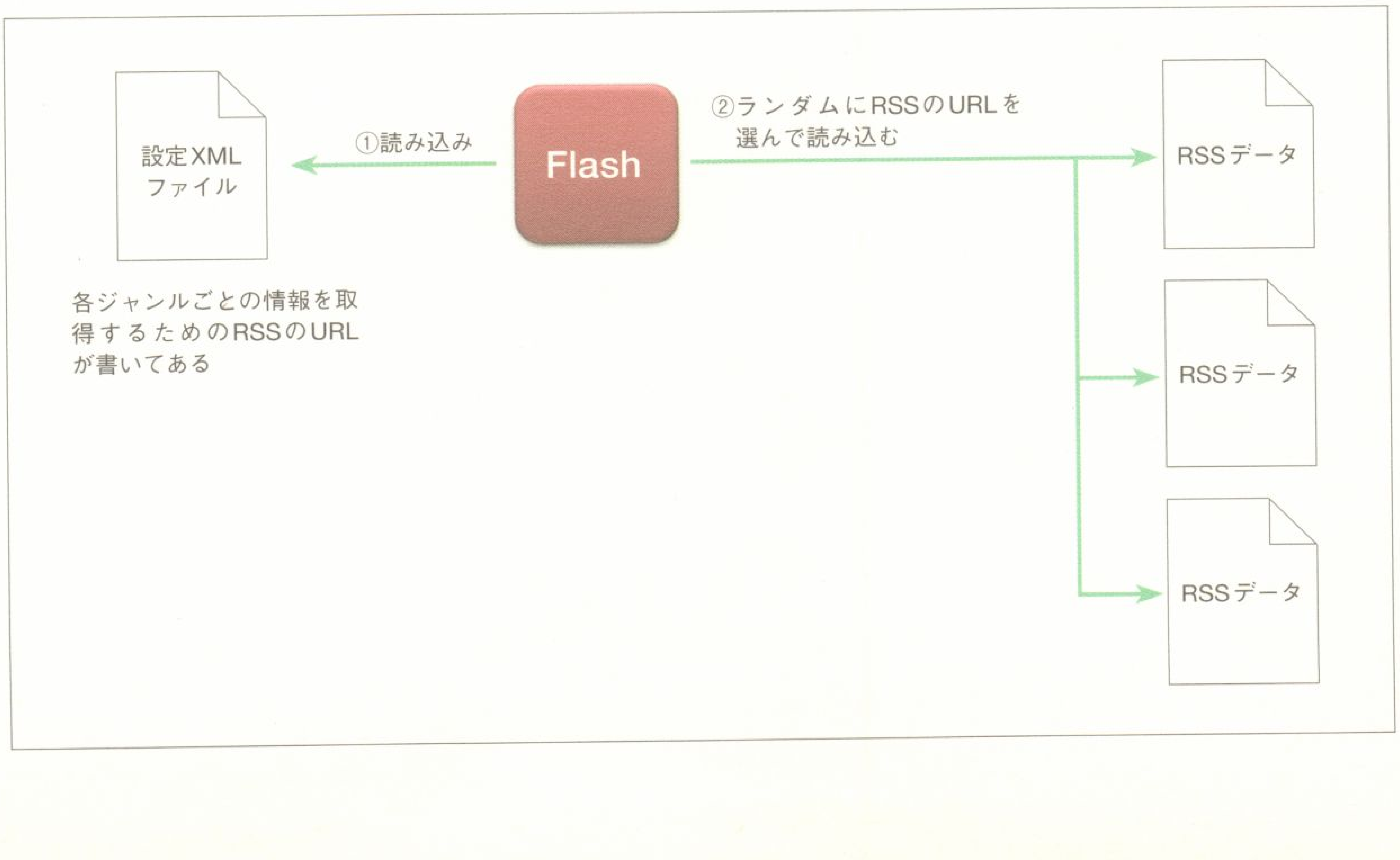
  // 吹き出しのムービークリップすべてを非表示にする
  for(var j = 1; j <= 5; j++) {
    var fukidashiName = "fukidashi" + j + "_mc";
    var fukidashi_mc = container_mc.getChildByName(fukidashiName);
    fukidashi_mc.visible = false;
    fukidashi_mc.stop();
  }

  // XMLの読み込み開始
  loadXML();
}
```



STEP 2 設定XMLファイルを読み込む

RSS情報を記載している設定XMLファイルは、すべての動作に先だって行う必要があります。この設定XMLファイルのXML構造はこちらで自由に決めることができるので、できるだけActionScriptで読みやすい構造にしておくと良いでしょう。設定情報を外部ファイルにしておくことで、そこに書いてある変更であれば、例え後々値を変更しないといけなくなったとしても、再パブリッシュの必要は無くなります。



1

読み込むXMLは、RSSごとに1つのノードを作り、RSSのURLのほかに、右側に表示する大きいキャラクターのSWFファイル名や画面下に表示する小さいキャラクターの種類、吹き出しの形、情報の最後に付ける語尾などを決めています。つまり、これらの情報であれば、再パブリッシュの必要なく変更が可能です。

```
<?xml version="1.0" encoding="UTF-8"?>
<setting>
  <data bigchara="geino.swf" smallchara="huho,seiji,fashion,gourmet" fukidashi="1" news="http://ge-nou.net/
    index.xml"><![CDATA[だって、奥さん！]]></data>
  <data bigchara="fuho.swf" smallchara="geino,seiji,fashion,gourmet" fukidashi="2" news="http://projects.
    dynalogue.com/dev/cshool/conv.php?url=http://news.ceek.jp/rss/obituaries.rdf&enc=euc-jp"><![CDATA[…
    ご愁傷様…]]></data>
  <data bigchara="seiji.swf" smallchara="geino,huho,fashion,gourmet" fukidashi="3" news="http://projects.
    dynalogue.com/dev/cshool/conv.php?url=http://news.ceek.jp/rss/politics.rdf&enc=euc-jp"><![CDATA[ご存
    知ざます？]]></data>
  <data bigchara="fashion.swf" smallchara="geino,huho,seiji,gourmet" fukidashi="4" news="http://projects.
    dynalogue.com/dev/cshool/conv.php?url=http://woman.excite.co.jp/beauty/xml/?f=topics&enc=SJIS-
    win"><![CDATA[が、美しさのヒミツ]]></data>
  <data bigchara="gourmet.swf" smallchara="geino,huho,seiji,fashion" fukidashi="5" news="http://www.blog.
    s-woman.net/sweets/index.rdf"><![CDATA[激ウマですのよ〜]]></data>
  <data bigchara="akiba.swf" smallchara="geino,huho,seiji,fashion,gourmet" fukidashi="2" news="http://akiba.
    kakaku.com/index.xml"><![CDATA[ナリよ。]]></data>
  <data bigchara="mametisiki.swf" smallchara="geino,huho,seiji,fashion,gourmet" fukidashi="4" news="http://
    turenet.blog91.fc2.com/?xml"><![CDATA[なんだヨ、ハニー]]></data>
</setting>
```

2

XMLファイルやプレーンテキストファイルのような、テキストエディタで開くことができるファイルを総称して「テキストファイル」と呼びます。ActionScript 3.0でテキストファイルを読み込むときは、URLLoaderクラスを使うのが一般的です。URLLoader型のインスタンスはファイルの読み込みが終わるとCOMPLETEイベントを発行するので、そのイベントに対して関数をリスナ登録することで、読み込んだ後の動作を行うことができます。

■ URLLoader ································ P162へ
■ XMLの解析 ································ P165へ

```
/**
 * XMLを読み込む
 */
function loadXML()
{
   // URLLoader型のインスタンスを生成
   var xmlLoader:URLLoader = new URLLoader();
   // 読み込むファイルの種類を指定
   xmlLoader.dataFormat = URLLoaderDataFormat.TEXT;
   // 読み込み完了時に発行されるイベントを取得できるようにリスナ登録
   xmlLoader.addEventListener(Event.COMPLETE, xmlLoader_completeHandler);

   // URLRequestインスタンスを生成
   var req:URLRequest = new URLRequest("setting.xml");
   // XMLファイルを読み込む
   xmlLoader.load(req);
}
```

3 XMLファイルの読み込みや画像の読み込みなど、SWFファイル外部の情報を読み込む場合、「読み込むまでの処理」と「読み込んだ後の処理」を別の処理、つまり別々の関数と考えます。XMLファイルを読み込んだ後の処理では、XMLファイル内に記載されている情報を解析して、必要な情報を取得します。取得した情報はあとで利用するので、最初に定義した配列settingDataArrayに、dataノードごとにObject型データにまとめて、順番に格納します。

■ Object ………………………………… P128へ

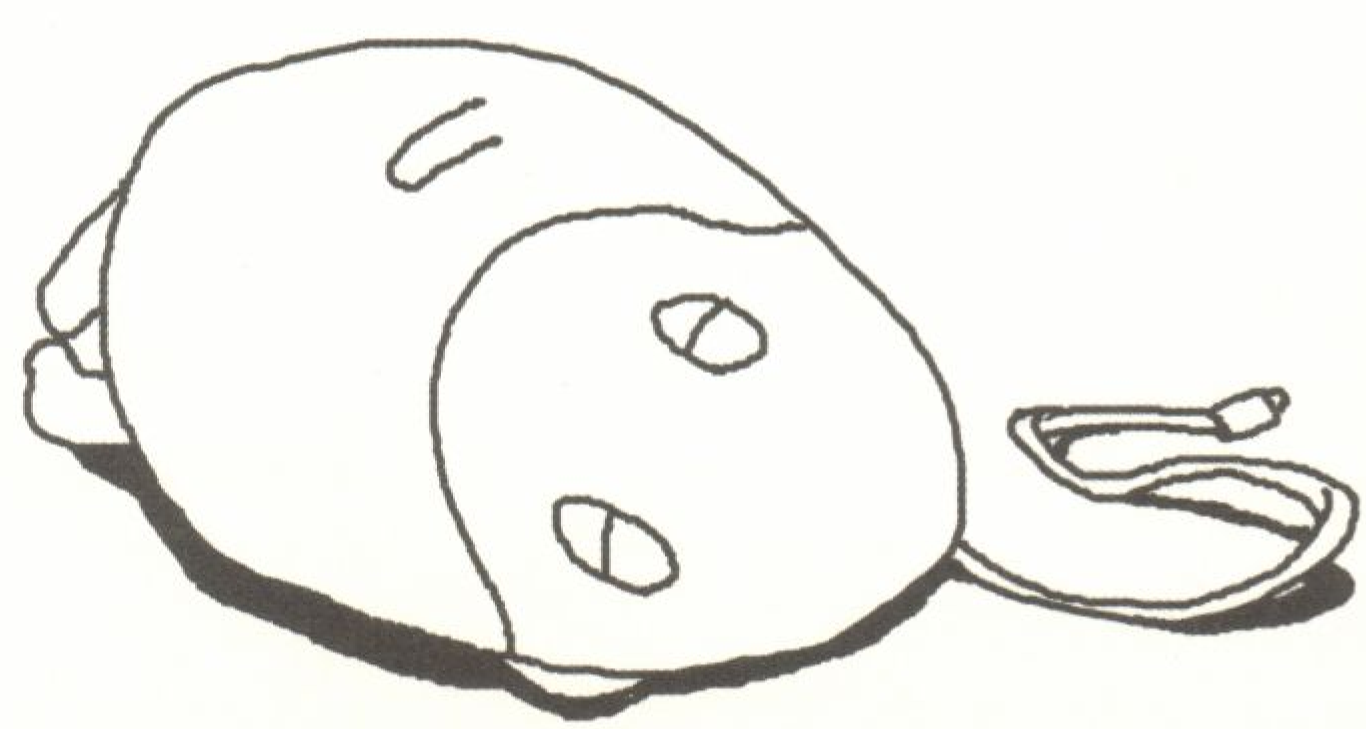
```
/**
 * XMLの読み込みが完了したときに呼び出される
 */
function xmlLoader_completeHandler(in_e)
{
    // 読み込んだテキストデータをXML解析できるようにXML型インスタンス生成に使用
    var rootXml:XML = new XML((in_e.target as URLLoader).data);

    // XML内のすべてのdataノードそれぞれについて解析処理
    for each(var dataXML in rootXml.data)
    {
        // データを保持するObject型インスタンスを生成
        var item = {};

        // 大きなキャラクターのSWFファイル名
        item.bigChara   = dataXML.@bigchara.toString();
        // 表示する小さなキャラクターの情報
        item.smallChara = dataXML.@smallchara.toString().split(",");
        // 吹き出しの種類情報
        item.fukidashi  = dataXML.@fukidashi.toString();
        // RSSのURL
        item.news       = dataXML.@news.toString();
        // 語尾に使用する文字列
        item.gobi       = dataXML.text().toString();
        // エントリー(記事)データ配列をあとで格納するための前準備
        item.entryTitle = null;

        // データを配列要素として最後尾に追加登録
        settingDataArray.push(item);
    }

    // データの処理が終わったので、次の表示するRSSを選択する処置に移る
    selectRss();
}
```



STEP 3 RSS情報を読み込む

RSS情報もXMLファイルの一種、つまりテキストファイルなので、URLLoaderクラスを用いて読み込みできます。RSS情報を利用するときのポイントは、「解析」です。RSSは種類によってXML構造が変わるため、それぞれ種類ごとに解析方法を変えていかないといけません。

1 STEP 2と同様に、URLLoaderクラスを使ってXML情報を読み込みます。URLLoaderクラスを使う部分は設定XMLファイルの読み込み方法とほとんど変わりませんが、ここでは設定ファイルに記載されていた複数のRSSの中で、どれを使用するかランダムに決めているプログラムが追加されているところと、すでに読み込んでいるRSSであれば、読み込み処理を省略する処理が追加されています。

```
/**
 * 表示するRSSをランダムに決定し、必要に応じて読み込む
 */
function selectRss()
{
   // ランダムにRSS情報を決定
   // RSS種類はsettingDataArrayの要素の数だけある
   selectedRssNum = Math.floor(Math.random() * settingDataArray.length);

   // もし決定したRSSがまだ読み込まれていない場合
   if(settingDataArray[selectedRssNum].entryTitle == null)
   {
      // RSSを読み込むための処理
      var rssXmlLoader:URLLoader = new URLLoader();
      rssXmlLoader.dataFormat = URLLoaderDataFormat.TEXT;
      rssXmlLoader.addEventListener(Event.COMPLETE, rssXmlLoader_completeHandler);

      var rssReq:URLRequest = new URLRequest(settingDataArray[selectedRssNum].news);
      rssXmlLoader.load(rssReq);
   }
   // 決定したRSSがすでに読み込まれている場合
   else
   {
      // すでにRSSデータはあるのでアニメーションを開始
      startAnimating();
   }
}
```

2 今回は3種類のRSSに対応できるようなプログラムを考えていきましょう。それぞれRSSのXML構造には特徴的な部分があるので、そこからまずRSS種類を判別し、次にそれぞれの種類に合った解析方法でプログラムを書いていきます。

名称	RSS 0.92 / 2.0 Really Simple Syndication	RSS 1.0 RDF Site Summary	Atom
判別方法	ルートノードの名前が「rss」	ルートノードの名前が「RDF」	ルートノードの名前が「feed」
記事タイトルへのアクセス	itemノードの子ノードtitle（itemノードがルートノードの直下ではない）	itemノードの子ノードtitle	entryノードの子ノードtitle
名前空間		名前空間を指定すること http://purl.org/rss/1.0/	名前空間を指定すること http://www.w3.org/2005/Atom

次のXMLはRSS1.0形式で出力されたRSS例です。
どのような構造になるのか、解析前にチェックしておきましょう。

ファイル(F)　編集(E)　表示(V)　ヘルプ(H)

```xml
<?xml version="1.0" encoding="utf-8"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:cc="http://web.resource.org/cc/"
  xmlns="http://purl.org/rss/1.0/">

<channel rdf:about="http://www.blog.s-woman.net/sweets/">
<title>[お取り寄せ]-うれしいおいしいお取り寄せ-  Happy Goody Please！</title>
<link>http://www.blog.s-woman.net/sweets/</link>
<description></description>
<dc:language>ja-JP</dc:language>
<dc:creator></dc:creator>
<dc:date>2008-06-18T00:00:00+09:00</dc:date>
<admin:generatorAgent rdf:resource="http://www.typepad.com/?v=1.0" />


<items>
<rdf:Seq><rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-cade.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/05/post-c842.html" />
</rdf:Seq>
</items>

</channel>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html">
<title>和三盆糖と小豆の香りにほっとする、風流な蒸菓子。「冨士屋の小男鹿」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html</link>
<description>長らく続いたこの連載も、今回で最後。記念すべき最終回は、少々地味ながら風雅な香り...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-18T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html">
<title>さっぱり、すっきり。柚子の香りに癒される、「茜庵の柚子じゅうす」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html</link>
<description>梅雨が始まり、湿気が多くてうっとうしい日々が続きます。ということで、前回に引き続...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-11T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-cade.html">
<title>アイスクリームにかけても美味！　「コクテール堂のコーヒーシロップ」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-cade.html</link>
<description>このコラムでもたびたび紹介しているように、“コーヒーもの”に目がない私。またまた...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-04T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/05/post-c842.html">
<title>素材がいいから、いくらでも食べられる！　「富士見堂のカレー煎餅としらす揚げ」</title>
```

行 34, 列 71

ルートノード

RSS全体の情報

記事１つ分の情報

記事１つ分の情報

記事１つ分の情報

Itemノードが周期的に続く

PRACTICE 6

3 RSSに含まれる情報はいろいろあります。その中で今回必要なのは記事のタイトルなので、それを解析し取得するプログラムをRSS種類ごとに書いています。そして、取得したタイトル情報は配列settingDataArrayの情報の1つ、配列entryTitleに追加していきます。

```
/**
 * RSSの読み込みが完了したときに呼び出される
 * RSSを解析し、データを読み込む
 */
function rssXmlLoader_completeHandler(in_e)
{
   var rssXml:XML = new XML(in_e.target.data);

   settingDataArray[selectedRssNum].entryTitle = [];

   // RSS 0.9/2.0 形式(Really Simple Syndication)の場合
   if(rssXml.localName() == "rss")
   {
      for each(var p in rssXml..item)
      {
         settingDataArray[selectedRssNum].entryTitle.push( p.title.text().toString() );
      }
   }
   // RSS 1.0 形式(RDF Site Summary)の場合
   else if(rssXml.localName() == "RDF")
   {
      default xml namespace = new Namespace("http://purl.org/rss/1.0/");

      for each(var q in rssXml.item)
      {
         settingDataArray[selectedRssNum].entryTitle.push( q.title.text().toString() );
      }
   }
   // atom 形式の場合
   else if(rssXml.localName() == "feed")
   {
      default xml namespace = new Namespace("http://www.w3.org/2005/Atom");

      for each(var r in rssXml.entry)
      {
         settingDataArray[selectedRssNum].entryTitle.push( r.title.text().toString() );
      }
   }

   // 解析した結果、記事タイトルが1件以上取得できた場合
   if(settingDataArray[selectedRssNum].entryTitle.length > 0)
   {
      // アニメーションを開始する
      startAnimating();
   }
   // 記事タイトルが取得できなかった場合
   else
   {
      // 他のRSSを読み込むため、再度RSSをランダムに決定
      selectRss();
   }
}
```

STEP 4 アニメーションを組み立てる

いろいろなパーツを組み合わせて1つのアニメーションを作成するときの1つの方法として、フレームでコントロールする方法があります。この方法は非常にシンプルで、かつタイムライン上のアニメーションを用いている場合に便利な手法です。今回もENTER_FRAMEイベントを使って、フレーム数で各キャラクターの動きや吹き出しの動きを制御します。

1 データの取得作業が終わったら、いよいよキャラクターや吹き出しを表示させる処理、アニメーションを実行します。タイミングをとりながら複数の要素のアニメーションを実行させていく手法は数々ありますが、今回はENTER_FRAMEイベントを使って、イベント取得開始を0フレームとし、以降決まったフレームでアニメーションを実行していくことで演出をコントロールしていきます。



2 アニメーションを開始するときにはENTER_FRAMEイベントをaddEventListener( )すると同時に、カウンタ変数を0にする（リセットする）ことを忘れずにしましょう。

```
/**
 * アニメーションを開始
 */
function startAnimating()
{
   // カウントを0に戻す
   enterFrameCount = 0;

   // アニメーション制御用のENTER_FRAMEイベントを取得できるようにリスナ登録
   addEventListener(Event.ENTER_FRAME, mainProcess_enterFrameHandler);
}
```

3

ENTER_FRAMEイベントによって呼び出される関数では、呼び出す度にカウンタ変数を1ずつ足し算していくことで、現在何フレームかを取得できるようにするのがポイントです。あとはif文で、決められたフレーム数で実行する関数を決めていきます。

```
/**
 * アニメーション制御用ENTER_FRAMEイベントによって実行されるリスナー関数
 */
function mainProcess_enterFrameHandler(in_e)
{
  if(enterFrameCount == 5)
  {
    // 大きなキャラクターのSWFファイルを読み込む処理を開始
    loadCharaSwf();
  }
  else if(enterFrameCount == 10)
  {
    // 小さいキャラクターを配置し、アニメーション開始
    createSmallChara();
  }
  else if(enterFrameCount == 20)
  {
    // 吹き出しを表示
    createFukidashi();
  }
  else if(enterFrameCount == 25)
  {
    // セリフを表示
    showLine();
  }
  else if(enterFrameCount == 75)
  {
    //「あらま。」の吹き出しを表示
    createSmallFukidashi();
    // キャラクターの動きを止める
    stopCharaAnimation();
  }
  else if(enterFrameCount == 130)
  {
    // すべての表示要素に対して消えるアニメーションを開始
    hideAllCharaAnimation();
  }

  * 略 *

  // カウントアップ
  enterFrameCount++;
}
```

4

アニメーションが一通り終了するタイミングで、次の処理、つまり次のRSS情報を取得するところに処理をつなげる必要があります。
また、表示要素のインスタンスを消したり、一旦ENTER_FRAMEイベントのリスナーを解除する処理も書きます。

```
function mainProcess_enterFrameHandler(in_e)
{

  * 略 *

  else if(enterFrameCount == 145)
  {
    // アニメーション制御用のENTER_FRAMEイベントを一旦リスナー解除する
    removeEventListener(Event.ENTER_FRAME, mainProcess_
      enterFrameHandler);
    // 表示要素を初期状態に戻す
    reset();
    // 次のニュースを選択する
    selectRss();
  }

  * 略 *

}
```

STEP 5 アニメーションを実行する

フレームごとに実行される関数の中で、それぞれのパーツに対して実際のアニメーションを行う処理を記述します。各処理を別々に作成していくことができるようになっているので、シンプルに考えることができます。

1 ENTER_FRAMEイベントによって制御されたアニメーションのうち、大きなキャラクターは外部SWFを読み込んで表示させています。外部SWFファイルを読み込むときはLoaderクラスを使用します。Loaderクラスのインスタンスは MovieClip インスタンスを同じように、addChild( )にて表示させることができます。ちなみに、読み込む外部SWFの名前は設定ファイルから取得しています。

■ Loader …………………… P158へ
■ addChild ……………… P124へ

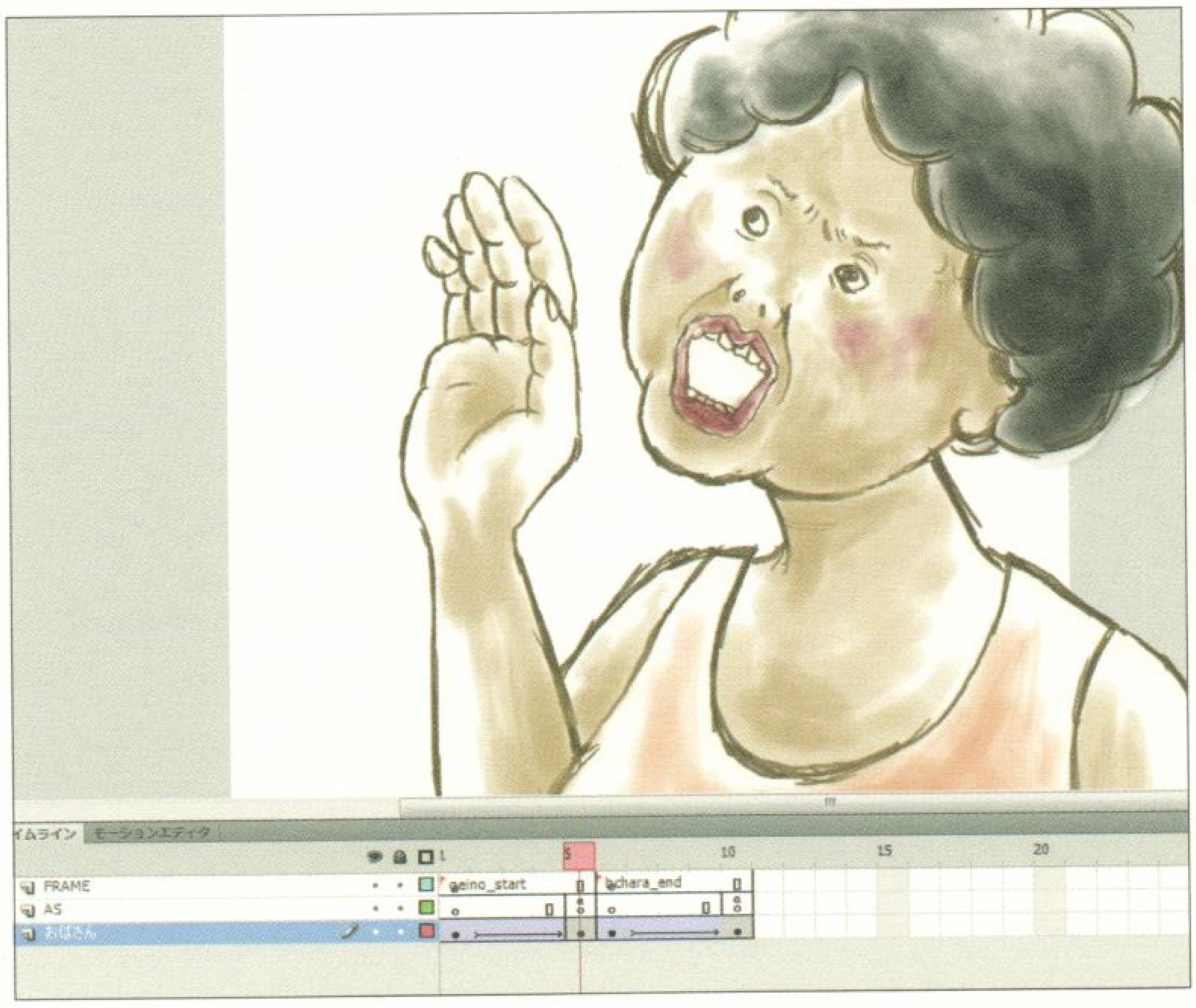
```
/**
 * 大きなキャラクターを表示する
 * 外部SWFファイルを読み込む
 */
function loadCharaSwf()
{
    // Loaderクラスからインスタンスを生成
    var bigCharaLoader:Loader = new Loader();
    bigCharaLoader.x = 380;
    // あとでインスタンスを取得するため、インスタンス名を設定
    bigCharaLoader.name = "bigChara";

    // container_mc配下に加える
    container_mc.addChild(bigCharaLoader);

    // 外部SWFのURL
    var bigCharaSwfUrl = settingDataArray[selectedRssNum].bigChara;

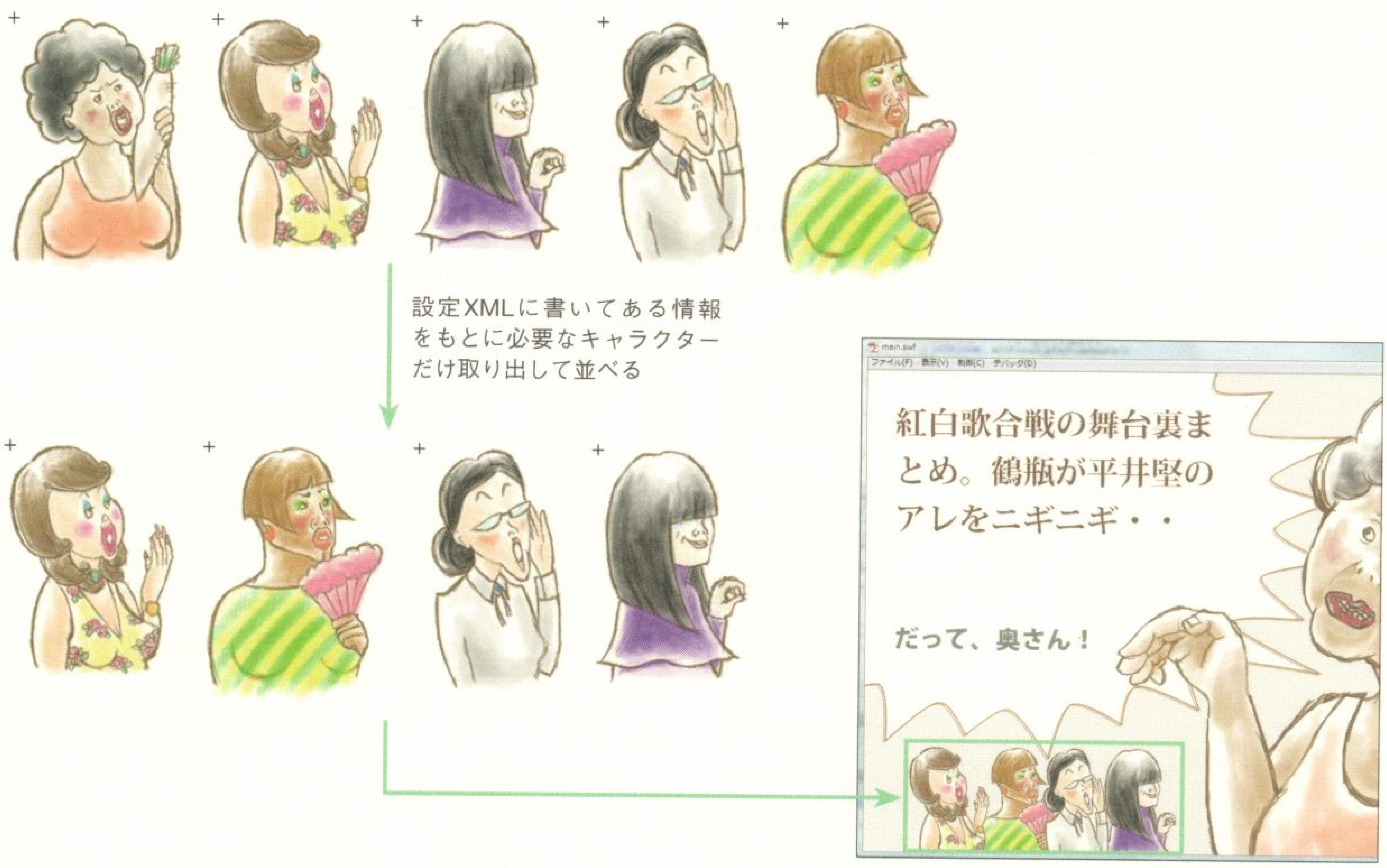
    // 外部SWFを読み込み
    bigCharaLoader.load(new URLRequest(bigCharaSwfUrl));
}
```

2 読み込まれる外部SWFファイルには、表示用のアニメーションと、消えるときのアニメーションを作っておきます。表示用のアニメーションは読み込みと同時に実行されるのですが、消えるときのアニメーションは、gotoAndPlay( )で命令する必要があります。そのときは、Loaderインスタンスのcontentプロパティが外部SWFファイルのステージに該当するので、それに対してgotoAndPlay( )を命令します。




```
// 大きなキャラクターを消すアニメーションを開始
(container_mc.getChildByName("bigChara") as Loader).content.gotoAndPlay("bchara_end");
```

3 画面下に表示される小さなキャラクターはすでにcontainer_mcの中に置いているので、必要に応じて非表示になっているものを表示状態にして使用します。設定XMLファイルのsmallcharaに書かれていた内容と、container_mcの中に配置したムービークリップのインスタンス名を使って(XMLに記述されている情報から簡単に指定できるインスタンス名にしておくことで)、for文の中の処理をより単純にするのがポイントです。
また、表示するムービークリップは等間隔で横に並べ、表示と同時に1フレームから再生するようにします。




```
/**
 * 小さいキャラクターを表示する
 * 設定XML情報のsmallcharaの値で表示するキャラクターを決定する
 */
function createSmallChara()
{
    // 表示するキャラクターの情報の数を取得
    var m = settingDataArray[selectedRssNum].smallChara.length;

    for (var j = 0; j < m; j++)
    {
        // すでに配置してある小さなキャラクターのインスタンス名を作成
        var mcName = settingDataArray[selectedRssNum].smallChara[j] + "_mc";
        ⬆ 例：geino_mc
        // インスタンス名からムービークリップの参照を取得
        var mc1 = container_mc.getChildByName(mcName);
        // 座標位置を設定
        mc1.x = -100 * j +370;
        mc1.y = 600;
        // タイムラインアニメーションを開始
        mc1.gotoAndPlay(1);
        // 表示状態に設定
        mc1.visible = true;
    }
}
```

4 テキストフィールドの下に表示される吹き出しも、考え方は小さなキャラクターのときと同じです。設定XMLファイルのfukidashiに書かれている内容から、簡単にcontainer_mcの中の吹き出しムービークリップのインスタンス名を指定できるようにしておけば、プログラムを短くすることができます。

```
/**
 * 設定XML情報のfukidashiの値をもとに吹き出しを表示する
 */
function createFukidashi()
{
  // すでに配置してある吹き出しムービークリップのインスタンス名を作成
  var mcName = "fukidashi" + settingDataArray[selectedRssNum].fukidashi + "_mc";
  ↑ 例：fukidashi1_mc～fukidashi5_mc
  // インスタンス名からムービークリップの参照を取得
  var mc = container_mc.getChildByName(mcName);
  // タイムラインアニメーションを開始
  mc.gotoAndPlay(1);
  // 表示状態に設定
  mc.visible = true;
}
```

COLUMN

文字コードに対する対応

私たちは普段扱っている日本語などの文字のデータには種類があります。同じ「あ」という文字でも、種類によって実際のデータは違います。その文字データの種類のことを「文字コード」と呼んでいます。
たとえば、私たちが普段から使っているWindowsやMac OS Xでは、「Shift JIS」という種類の文字コードが標準で使われています。一方で、最近のWebコンテンツでは「UTF-8」という種類の文字コードで制作することが一般的になりつつあります。
Flashでは、基本的に「UTF-8」が標準の文字コードです。ですので、「UTF-8」形式のテキストデータであれば問題なく読み込めますが、それ以外の文字コードの場合は、うまく読み込んでくれない可能性があります。そんなときに覚えてほしいのが、次の1行のプログラムです。

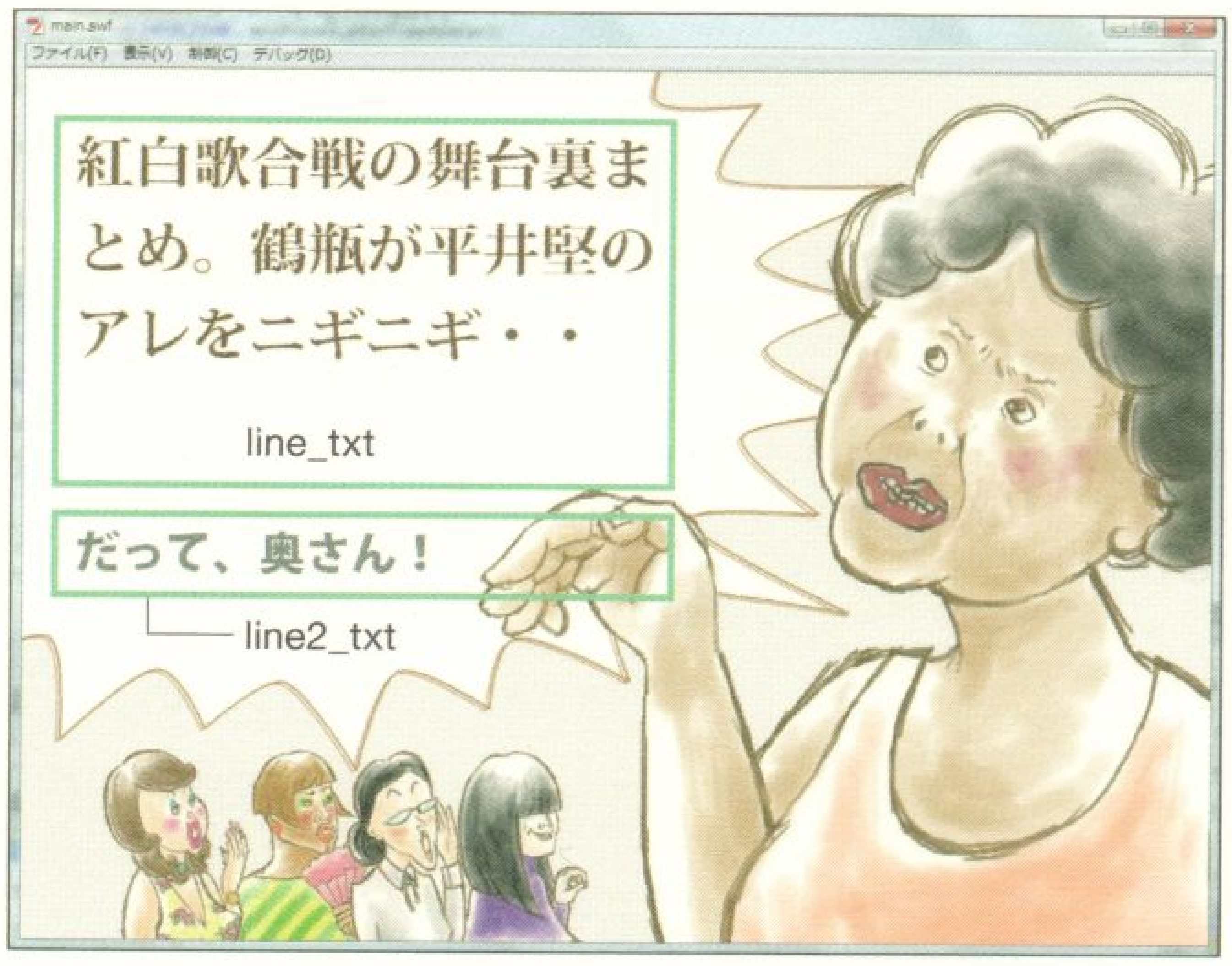
```
System.useCodePage = true;  ← 指定しないと通常はfalse
```

このプログラムを1行書けば、そのSWFファイルはSWFを実行している環境のOSの通常の文字コードでテキストデータを解釈しようとします。具体的に言うと、SWFを実行したOSが日本語環境であれば、Shift JISの文字コードで文字データを読み込んでくれるようになります(その代わり、UTF-8の文字コードは正常に読むことはできません)。
しかし、外部ファイルを読み込む処理の場合、読み込んだあとで文字コードの判別をして上記プログラムを実行するのは至難の業であるため、通常はSWFが使用する文字コードを一本化します。ですので、今回のような複数のRSSを読み込むような場合は、RSSによって文字コードが違っている可能性があり、そのときは上記プログラムで対応するのはなかなか難しいことになります。
ですので、そのようなときにはサーバ側などでUTF-8に文字コードを変換した上でFlashでデータを読み込むなどの別作業を行ったほうが安全です。

STEP 6 吹き出しの中にタイトルを表示させる

RSSから取得したタイトル文字列を表示させます。
2つのテキストフィールドに別々の文字を表示させる点に注意しましょう。

1 タイトルを表示するのは、あらかじめ置いているテキストフィールドline_txtです。また、大きなキャラクターごとに別々の語尾を付けることができるようになっており、その言葉はline2_txtに表示するようになっています。



2 line_txtとline2_txtのtextプロパティに空文字を代入して空にしておき、次に前に決めたRSSの中で、どのタイトルを表示するか決めます。決め方は、まずRSSに含まれる記事数を取得し、その記事数内でランダムに記事番号を決定します。この記事番号は配列番号として使用することで、ランダムに記事タイトルが取得できます。

```
<items>
<rdf:Seq><rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/06/post-cade.html" />
<rdf:li rdf:resource="http://www.blog.s-woman.net/sweets/2008/05/post-c842.html" />
</rdf:Seq>
</items>

</channel>

<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html">
<title>和三盆糖と小豆の香りにほっとする、風流な蒸菓子。「冨士屋の小男鹿」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-0c16.html</link>
<description>長らく続いたこの連載も、今回で最後。記念すべき最終回は、少々地味ながら風雅な香り...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-18T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html">
<title>さっぱり、すっきり。柚子の香りに癒される、「茜庵の柚子じゅうす」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-8e60.html</link>
<description>梅雨が始まり、湿気が多くてうっとうしい日々が続きます。ということで、前回に引き続...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-11T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/06/post-cade.html">
<title>アイスクリームにかけても美味！　「コクテール堂のコーヒーシロップ」</title>
<link>http://www.blog.s-woman.net/sweets/2008/06/post-cade.html</link>
<description>このコラムでもたびたび紹介しているように、“コーヒーもの”に目がない私。またまた...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-06-04T00:00:00+09:00</dc:date>
</item>
<item rdf:about="http://www.blog.s-woman.net/sweets/2008/05/post-c842.html">
<title>素材がいいから、いくらでも食べられる！　「富士見堂のカレー煎餅としらす揚げ」</title>
<link>http://www.blog.s-woman.net/sweets/2008/05/post-c842.html</link>
<description>ケーキやチョコはしょっちゅうおやつに食べている私ですが、出来るだけ買わないように...</description>

<dc:subject>お取り寄せ</dc:subject>

<dc:creator>集英社 s-woman</dc:creator>
<dc:date>2008-05-28T00:00:00+09:00</dc:date>
</item>

</rdf:RDF>
行 15, 列 28
```



読み込んだRSS情報の記事の中から1つをランダムに選んでテキストフィールドに表示

```
/**
 * タイトルと語尾文字列を表示する
 */
function showLine()
{
    // RSS内の記事の数を調べる
    var entryLength = settingDataArray[selectedRssNum].entryTitle.length;

    // 1つの記事番号をランダムに取得
    selectedItemNum = Math.floor(Math.random() * entryLength);

    // 記事のタイトルを取得
    var targetTitle = settingDataArray[selectedRssNum].entryTitle[selectedItemNum];
    // テキストフィールドに表示
    line_txt.text = targetTitle;
}
```

3 語尾部分はline2_txtのtextプロパティに、設定XMLファイルに記載されていた語尾文字列をそのまま代入します。語尾文字列はRSSごとに設定XMLファイルに記載されているので、大きなキャラクターごとに違う語尾を付けることができます。

```
// 語尾文字列を取得
var targetGobi  = settingDataArray[selectedRssNum].gobi;
// テキストフィールドに表示
line2_txt.text = targetGobi;
```

COLUMN

動的に作ったインスタンスを削除する

Flashの処理スピードを上げたり、動作を安定化することを考えた場合、非常に重要なのが「不要になったインスタンスを削除する」という作業です。もし生成したインスタンスが不必要になった場合は、まずそれが表示系のもの(addChild( )して表示させたもの)であれば、removeChild( )します。そして、そのインスタンスが何かのイベントにリスナ登録していた場合は、そのリスナ登録もremoveEventListener( )で解除します。そして、必要に応じてインスタンス変数にnullを代入します。そうすることで、Flash Player側にこのインスタンスは不必要になったことを伝えることができ、あとはFlash Playerのタイミングでインスタンスが削除され、Flash Playerが使用するメモリ量が減ります。
ちなみに、上記のようなインスタンス管理は長期間動作を継続させるタイプのFlashコンテンツでは特に重要です。なぜなら、不必要なものを消していかないと、Flash Playerが必要とするメモリ量が永遠に増え続け、最悪フリーズなどの障害を引き起こしてしまうからです。

STEP7 繰り返し表示するためにアニメーションを終わらせる

繰り返し同じ動作をする処理を作る場合、繰り返し動作が一旦終わる度に「元の状態に戻す」、すなわちリセットするのが、非常に重要なポイントです。リセットが不十分だと、繰り返し動作していくうちに、問題が発生する原因になります。

1 すべての要素を表示し終わったら、「あらま。」と言う小さな吹き出しを表示します。表示方法は、動的にSmallFukidashiクラスのインスタンスを生成して配置し、addChild( )して表示するベーシックな方法です。



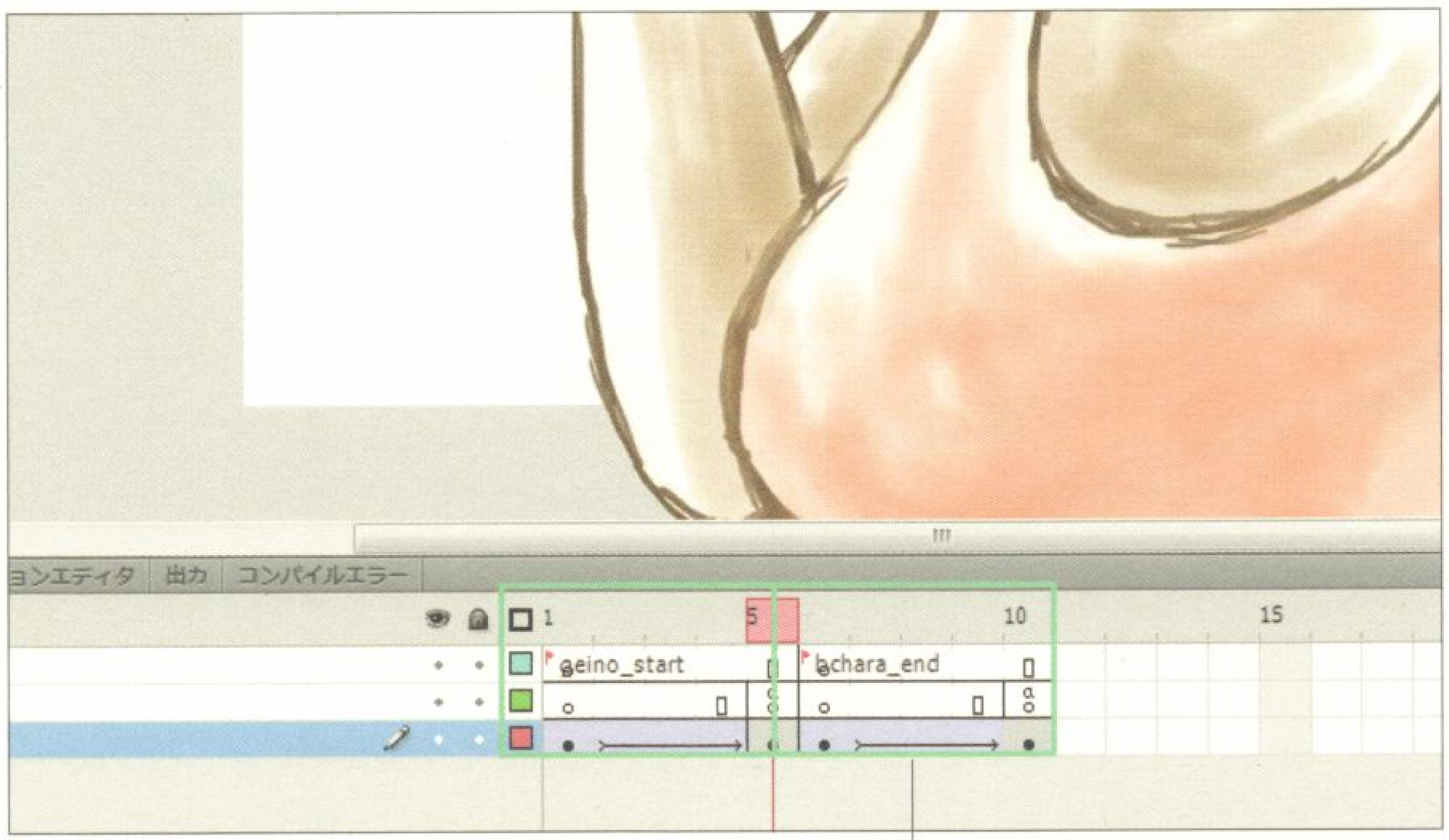

```
/**
 * 小さな吹き出しを表示する
 * インスタンスを都度生成して表示している
 */
function createSmallFukidashi()
{
    // リンケージ名：SmallFukidashiのインスタンスを生成
    var smallFukidashi = new SmallFukidashi();
    smallFukidashi.x = 18;
    smallFukidashi.y = 320;
    smallFukidashi.name = "smallFukidashi";
    // メインタイムラインに追加
    addChild(smallFukidashi);
}
```

2 小さな吹き出しの表示と同じタイミングで、大小すべてのキャラクターの小刻みなループアニメーションを止めます。外部SWFをLoaderクラスで読み込んだ大きなキャラクターをコントロールするときは、contentプロパティを通じて操作すること、また小さなキャラクターは設定XMLファイルで指定している対象キャラクターについてのみ処理を行います。また、それぞれのキャラクターの素材を作るときに、ラベル名を付けておくとあとでアニメーションの命令がしやすくなります。

```
/**
 * キャラクターのアニメーションを停止する
 */
function stopCharaAnimation()
{
    // 大きなキャラクターの動きを止める
    (container_mc.getChildByName("bigChara") as Loader).content.geinoobasan_mc.stop();

    // 小さなキャラクターの動きを止める
    var n = settingDataArray[selectedRssNum].smallChara.length;
    for (var i = 0; i < n; i++)
    {
        // 小さなキャラクターのムービークリップのインスタンス名を作成
        var mcName = settingDataArray[selectedRssNum].smallChara[i] + "_mc";
        // インスタンス名からムービークリップの参照を取得
        var mc = container_mc.getChildByName(mcName);
        // 停止するときのタイムラインに遷移
        mc.gotoAndStop("schara_stop");
    }
}
```

3 アニメーションを止めた後、一定時間後にすべての表示要素を消す処理を行います。ただしパッと消すのは寂しいので、それぞれのキャラクターのムービークリップに消えるときのアニメーションを用意しておき、そのアニメーションをgotoAndPlay( )で実行します。テキストフィールドについては、単にtextプロパティに空文字を代入して、文字を消しています。



geino.fla (geino.swf) 内の設定

消えるときのアニメーションが設定されているフレームとラベル(bchara_end)が、すべての外部読み込みSWFに同じように用意されている

```
/**
 * キャラクターや吹き出しなど画面上の要素を消すアニメーションを開始する
 */
function hideAllCharaAnimation()
{
   // 大きなキャラクターを消すアニメーションを開始
   (container_mc.getChildByName("bigChara") as Loader).content.gotoAndPlay("bchara_end");

   // 小さなキャラクターを消すアニメーション
   var n = settingDataArray[selectedRssNum].smallChara.length;
   for (var i = 0; i < n; i++)
   {
      // 小さなキャラクターのムービークリップのインスタンス名を作成
      var mcName = settingDataArray[selectedRssNum].smallChara[i] + "_mc";
      // インスタンス名からムービークリップの参照を取得
      var mc = container_mc.getChildByName(mcName);
      // 消えるときのタイムラインアニメーションを開始
      mc.gotoAndPlay("schara_end");
   }

   // すでに配置してある吹き出しムービークリップのインスタンス名を作成
   var fukidashiName = "fukidashi" + settingDataArray[selectedRssNum].fukidashi + "_mc";
   // インスタンス名からムービークリップの参照を取得
   var fukidashi_mc = container_mc.getChildByName(fukidashiName);
   // 大きな吹き出しを消すタイムラインアニメーションを開始
   fukidashi_mc.gotoAndPlay("bcharabal_end");

   // 小さな吹き出しを消すタイムラインアニメーションを開始
   getChildByName("smallFukidashi").gotoAndPlay("scharabal_end");

   // テキストフィールドに空文字を代入して表示文字を消す
   line_txt.text = "";
   line2_txt.text = "";
}
```

PRACTICE 6

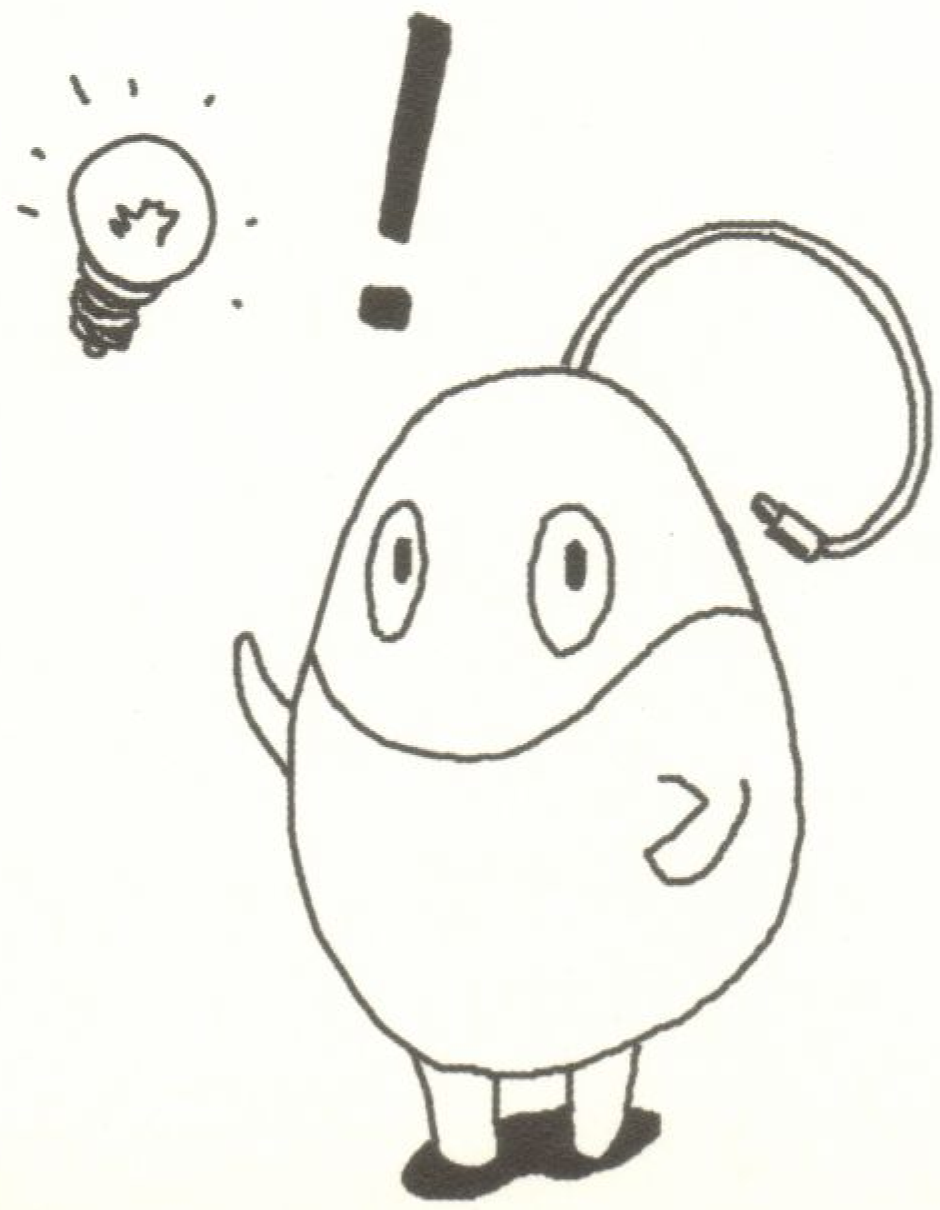
4 hideAllCharaAnimation( )を実行することで、画面上からはすべての要素が消えました。しかし、再度キャラクターを表示させる処理につなげるためには、それぞれの要素を最初の状態に戻しておく必要があります。この処理は、繰り返し動作するプログラムを考える上で非常に重要です。ステージ上に配置されているムービークリップ類はvisibleプロパティをfalseにし、動的に配置したムービークリップはremoveChild( )しておきます。

```
/**
 * 表示要素をアニメーション開始時の状態に戻す
 */
function reset()
{
  // 外部SWFファイルの情報をアンロード
  (container_mc.getChildByName("bigChara") as Loader).unload();
  // 大きなキャラクターを親ムービークリップから削除
  container_mc.removeChild(container_mc.getChildByName("bigChara"));

  // 小さなキャラクターで表示状態のものを非表示にする
  var n = settingDataArray[selectedRssNum].smallChara.length;
  for (var i = 0; i < n; i++)
  {
    // 小さなキャラクターのムービークリップのインスタンス名を作成
    var mcName = settingDataArray[selectedRssNum].smallChara[i] + "_mc";
    // インスタンス名からムービークリップの参照を取得
    var mc = container_mc.getChildByName(mcName);
    // 非表示にする
    mc.visible = false;
    // アニメーションを止め、1フレーム目に移動させておく
    mc.gotoAndStop(1);
  }

  // すでに配置してある吹き出しムービークリップのインスタンス名を作成
  var fukidashiName = "fukidashi" + settingDataArray[selectedRssNum].fukidashi + "_mc";
  // インスタンス名からムービークリップの参照を取得
  var fukidahsi_mc = container_mc.getChildByName(fukidashiName);
  // 吹き出しを非表示にする
  fukidahsi_mc.visible = false;
  // アニメーションを止め、1フレーム目に移動させておく
  fukidahsi_mc.gotoAndStop(1);

  // 小さな吹き出しをメインタイムラインから削除
  removeChild(getChildByName("smallFukidashi"));
}
```



STEP 8 スクリーンセーバーを作成する

最後にスクリーンセーバーを作成する手順を紹介します。スクリーンセーバー用SWFを作成する場合、基本的には通常のインターネット上で公開するSWF作成と変わらないですが、使用するツールによってFlashでできること以上の処理をサポートしていたり、逆に制約などが存在する場合もまったくないとは言い切れません。ですので、完成する前の途中段階から、こまめにスクリーンセーバー化してチェックを行っておくと良いでしょう。

1 SWFファイルを使ってスクリーンセーバーを作成する場合は、通常市販されているアプリケーションを使用します。現在主に使われているのは、「fla:ver」と「ScreenTime」というアプリケーションです。今回はfla:verを使ってスクリーンセーバーを作成します。

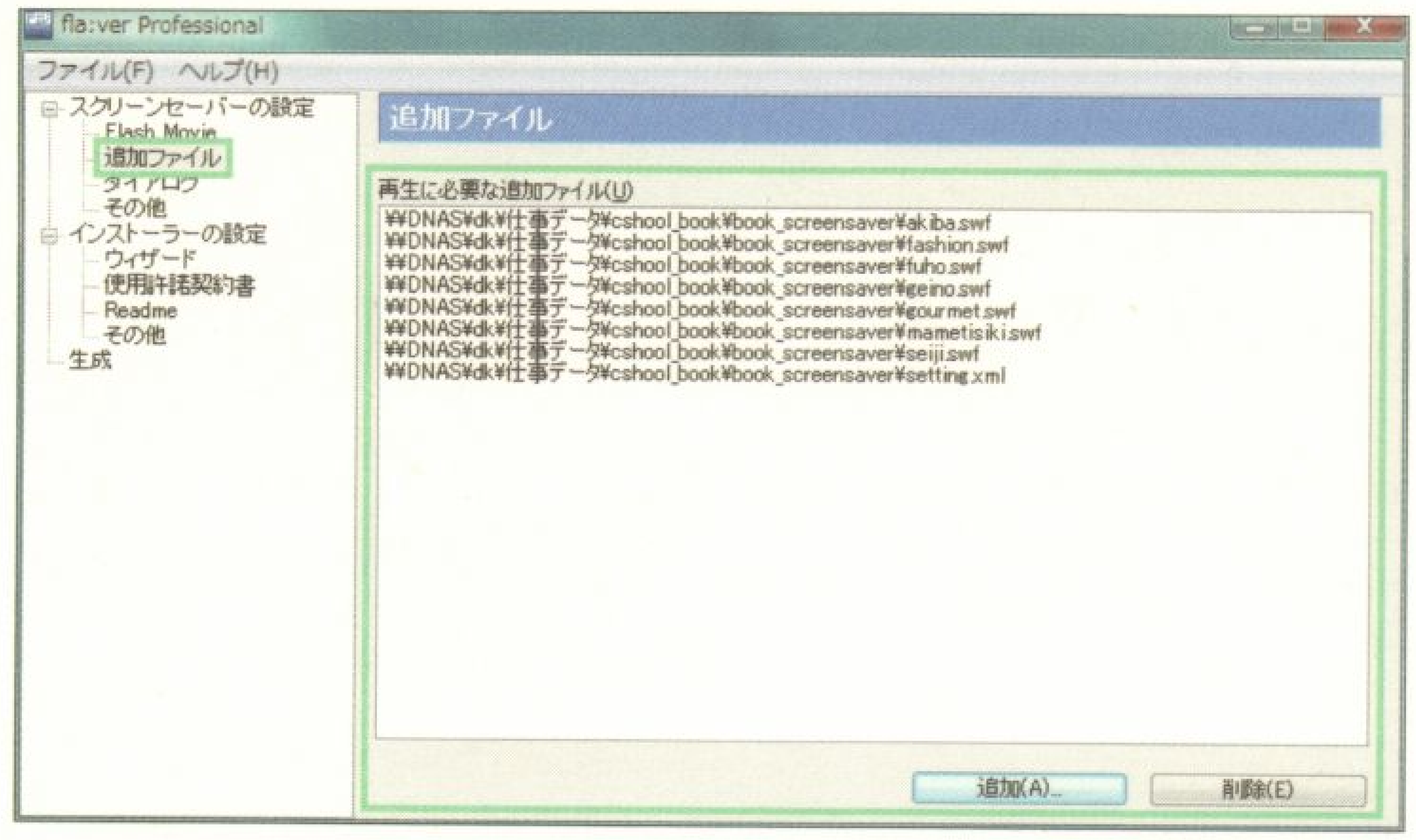
http://flaver.jp/

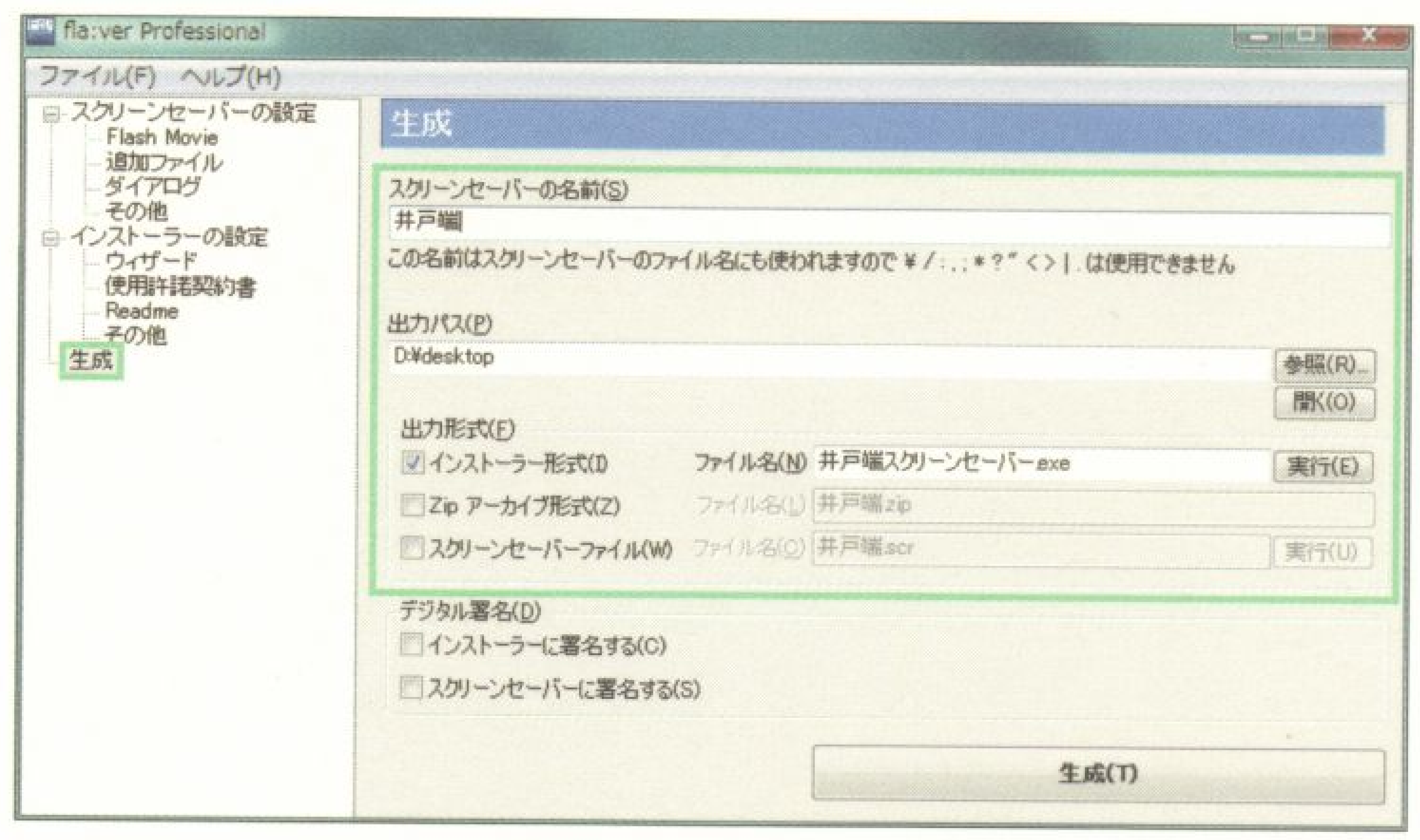


2 fla:verを起動したら、まず「Flash Movie」欄で本体に相当するSWFファイルを指定します。

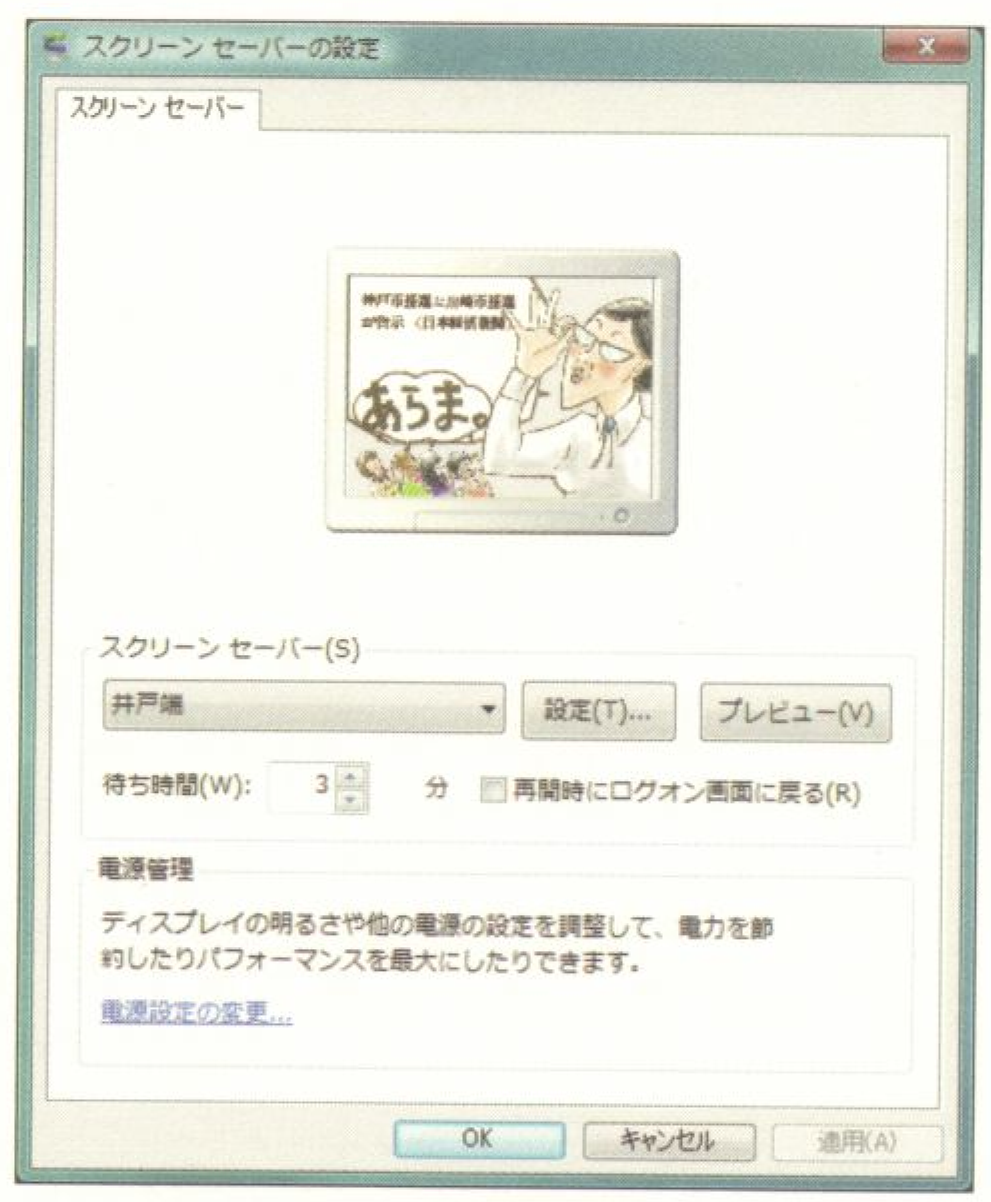


3 次に「追加ファイル」画面で、大きなキャラクターごとに存在する外部読み込みのSWFファイルすべてと、設定XMLファイルを指定しましょう。
あとは「生成」画面で名前やスクリーンセーバーファイルの名前を入力すれば、最低限の設定でスクリーンセーバーを作成することができます。





4 作成したスクリーンセーバーをインストールし、きちんと動くかどうかチェックを忘れないようにしましょう。
また、今回はWindows用のスクリーンセーバーを作成しましたが、Mac用もほぼ同様の手順で作成が可能です。別途Mac用のスクリーンセーバー作成ツールを用意する必要がありますが、配布する場合は両方のOSに対応しておくべきです。

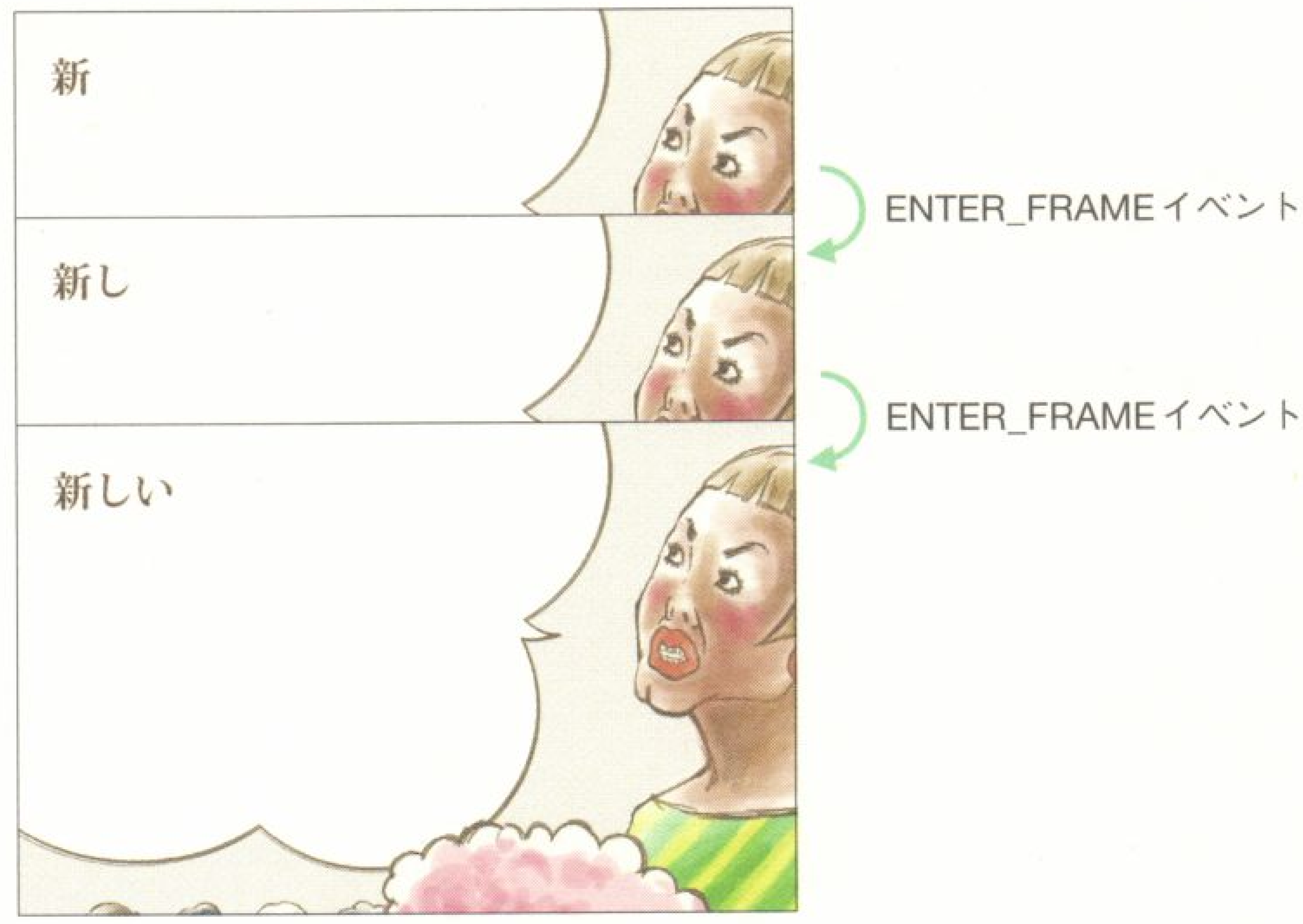


応 用

Ex.1 1文字ずつ文字を表示させる方法

今回のサンプルでは、パッと文章を見せるより、1文字ずつ文章を見せていく方が、よりキャラクターが喋っている雰囲気が出ます。その効果を付けてみましょう。

1 プログラムで表示する文章を頭から1文字ずつ一定時間ごとにテキストフィールドに加えていくだけで、1文字ずつ表示していくことができます。そしてその加えるタイミングですが、ENTER_FRAMEイベントを使用するのが一番適当でしょう。今回のサンプルでは、タイトル文字列と、語尾の文字列それぞれを1文字ずつ表示するだけで可能です。また、テキストフィールドに文字を追加していく場合、+=演算子ではなく、テキストフィールドインスタンスのappendText( )を使いましょう。




```
/**
 * タイトルと語尾文字列を表示する
 */
function showLine()
{
   // RSS内の記事の数を調べる
   var entryLength = settingDataArray[selectedRssNum].entryTitle.length;

   // 1つの記事番号をランダムに取得
   selectedItemNum = Math.floor(Math.random() * entryLength);

   // 1文字ずつタイトルを表示するためのENTER_FRAMEイベント取得開始
   addEventListener(Event.ENTER_FRAME, enterFrameHandler1);
}

/**
 * 1文字ずつタイトルを表示するためのENTER_FRAMEイベントハンドラ関数
 */
function enterFrameHandler1(in_e:Event):void
{
   // 表示するタイトル文字列
   var targetTitle = settingDataArray[selectedRssNum].entryTitle[selectedItemNum];

   // テキストフィールドに1文字加える
   line_txt.appendText(targetTitle.charAt(line_txt.text.length));

   * 以下略 *
}
```

PRACTICE 6

2 文字を1文字ずつ表示する場合、文章が完全に表示し終わるタイミングが文章の長さによって長くなったり短くなったりする、ということを注意しないといけません。ですので、文章が完全に表示し終わるタイミングをチェックして、表示し終わったら次の処理に加える、という流れでプログラムを組むほうが適切です。

```
/**
 * 1文字ずつタイトルを表示するためのENTER_FRAMEイベントハンドラ関数
 */
function enterFrameHandler1(in_e:Event):void
{
    // 表示するタイトル文字列
    var targetTitle = settingDataArray[selectedRssNum].entryTitle[selectedItemNum];

    // テキストフィールドに1文字加える
    line_txt.appendText(targetTitle.charAt(line_txt.text.length));

    // テキストフィールドに表示している文字列の長さが、表示するタイトル文字列以上の長さになったとき
    if (line_txt.text.length >= targetTitle.length){
        // このENTER_FRAMEイベントの取得を解除
        removeEventListener(Event.ENTER_FRAME, enterFrameHandler1);
        // 1文字ずつ語尾を表示するためのENTER_FRAMEイベントの取得を開始
        addEventListener(Event.ENTER_FRAME, enterFrameHandler2);
    }
}

/**
 * 1文字ずつ語尾文字列を表示するためのENTER_FRAMEイベントハンドラ関数
 */
function enterFrameHandler2(in_e):void
{
    // 表示する語尾文字列
    var targetGobi = settingDataArray[selectedRssNum].gobi;

    // テキストフィールドに1文字加える
    line2_txt.appendText(targetGobi.charAt(line2_txt.text.length));

    // テキストフィールドに表示している文字列の長さが、語尾文字列以上の長さになったとき
    if (line2_txt.text.length >= targetGobi.length){
        // このENTER_FRAMEイベントの取得を解除
        removeEventListener(Event.ENTER_FRAME, enterFrameHandler2);
        // 小さなキャラクター吹き出しを表示
        createSmallFukidashi();
    }
}
```

タイトル部分

語尾部分

応用

Ex.2 動的にクラス名を作って、インスタンスを生成する

動的にムービークリップを表示する場合、ActionScript 2.0ではリンケージ名という文字列とattachMovie( )という関数で、流れによって違うムービークリップを動的に生成するプログラムを比較的簡単に作ることができました。対してActionScript 3.0では、ムービークリップについてもnew演算子を使ってムービークリップを生成して動的にムービークリップを作ります。つまり、生成するムービークリップが動的に決まる場合は、クラス名を動的に作ることになります。

1 動的にクラス名を決め、そのインスタンスを生成する場合は、getDefinitionByName(クラス名)という関数を使用します。この関数でクラスのデータを持つ変数(インスタンス)を作り、その変数を使ってnew演算子で、動的に指定したクラスについてインスタンスを生成することができます。

間違い例

```
// 以下のような使い方はできない

// クラス名の文字列を生成
var smallCharaName = "SmallChara_Sample";

// クラスのインスタンス生成
var smallChara = new smallCharaName(); // smallCharaNameはクラスの名前であって、クラス情報を持ってはいない
```

正しい例

```
/**
 * 小さいキャラクターを表示する
 * 設定XML情報のsmallcharaの値で表示するキャラクターを決定する
 */
function createSmallChara()
{
   // 表示するキャラクターの情報の数を取得
   var n = settingDataArray[selectedRssNum].smallChara.length;
   for (var i = 0; i < n; i++)
   {
      // クラス名の文字列を生成
      var smallCharaID = "SmallChara_" + settingDataArray[selectedRssNum].smallChara[i];
      // クラス名の文字列からクラス情報を持ったインスタンスを生成
      var smallCharaClass = getDefinitionByName(smallCharaID) as Class;
      // クラス情報を持ったインスタンスからそのクラスのインスタンスを生成
      var smallChara = new smallCharaClass();
      smallChara.x = -100 * i +370;
      smallChara.y = 600;
      smallChara.name = smallCharaID;

      container_mc.addChildAt(smallChara, 0);
   }
}
```

そのほか応用例として以下のものがあります。

- 設定XMLファイルや大きなキャラクターのSWFをインターネット上に配置し、それをスクリーンセーバーが読み込む仕様にすることで、インストール後もRSSの内容を配布側でコントロールすることができる。

PRACTICE 7

プリローダーを使ってサイトを作る

表現力：★★★☆☆　プログラミング力：★★★☆☆

サイトをFlashで作る場合「ここをこうしたい」「こんな機能が欲しい」といった要望が次々と沸いてくるものです。このサンプルでは、上記のようなサイトづくりにおける、そんな「もう一工夫」の要素に焦点をあてました。実際の制作の現場でも特に要望の多い「プリローダーの作成」「外部SWFの制御」「ステージサイズの変更への対応」を実現する方法について、詳しく解説しています。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

- タイムラインを制御する………… P080へ
- MovieClipを制御する………… P083へ
- Loader………… P158へ
- 映像の扱い方………… P060へ
- ステージの情報を取得・設定する………… P144へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- プリローダーを作成する
- 外部SWFを制御する
- ステージサイズの変更に対応する

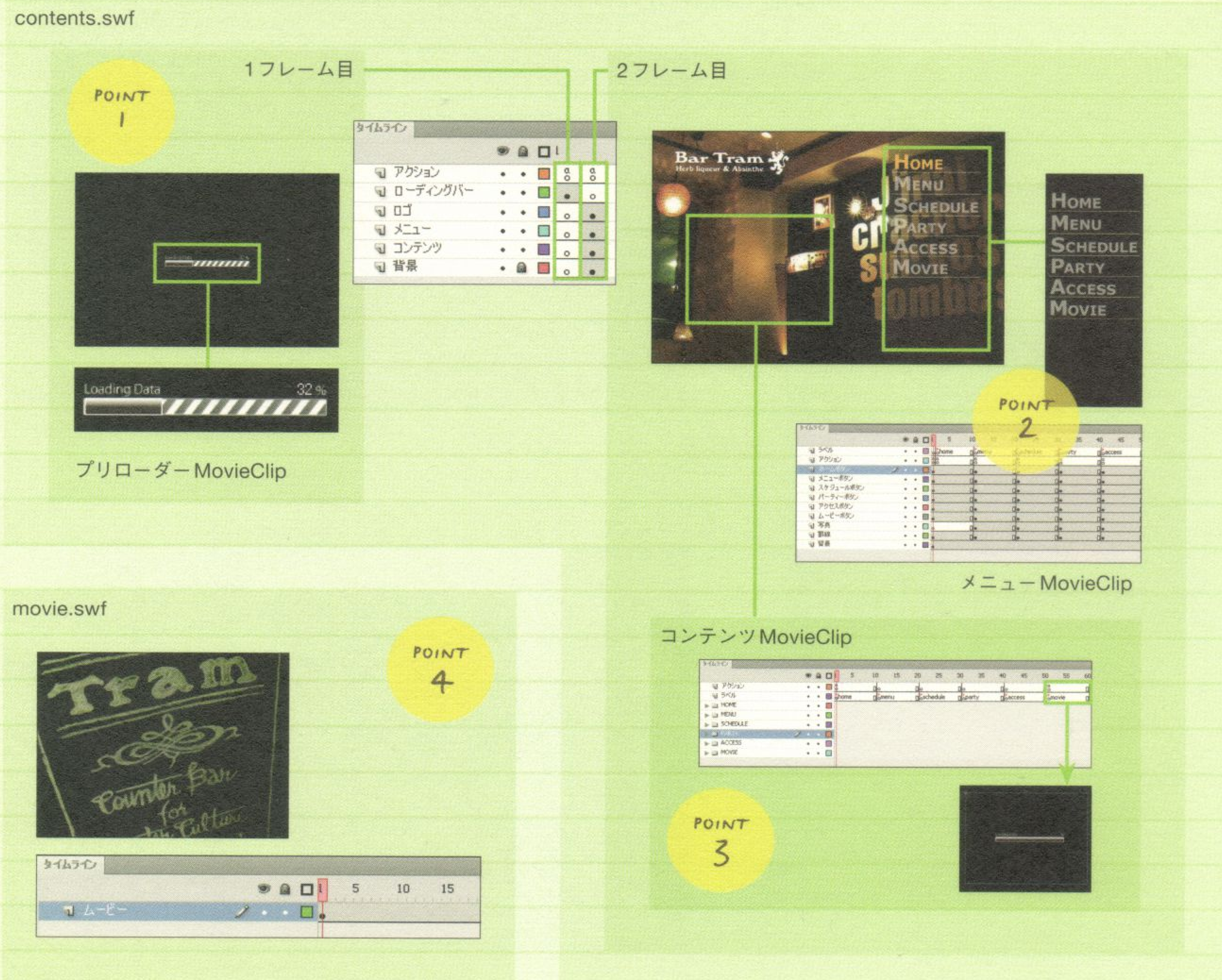
サンプルの説明



POINT

1	サイトを表示する前に、現在どれだけSWFファイルが読み込まれているかを表すプリローダーを表示します。
2	サイト内で再生するムービーを外部SWFとして用意しています。読み込んだSWFはActionScriptで制御できます。
3	コンテンツMovieClipのメニューはステージの大きさに合わせて、位置や大きさを変更します。

サンプルの構造



POINT

1 contents.swfのメインタイムラインの1フレーム目には、プリローダーのみが置かれています。実際のサイトのコンテンツMovieClipを2フレーム以降に置くことで、1フレーム目でcontents.swfのプリローダーを表示することが可能になります。

2 メニューのMovieClipは、各ページごとにタイムラインに分かれています。メインタイムライン上のfunction showContentを実行することで、メニューの表示が切り替わります。

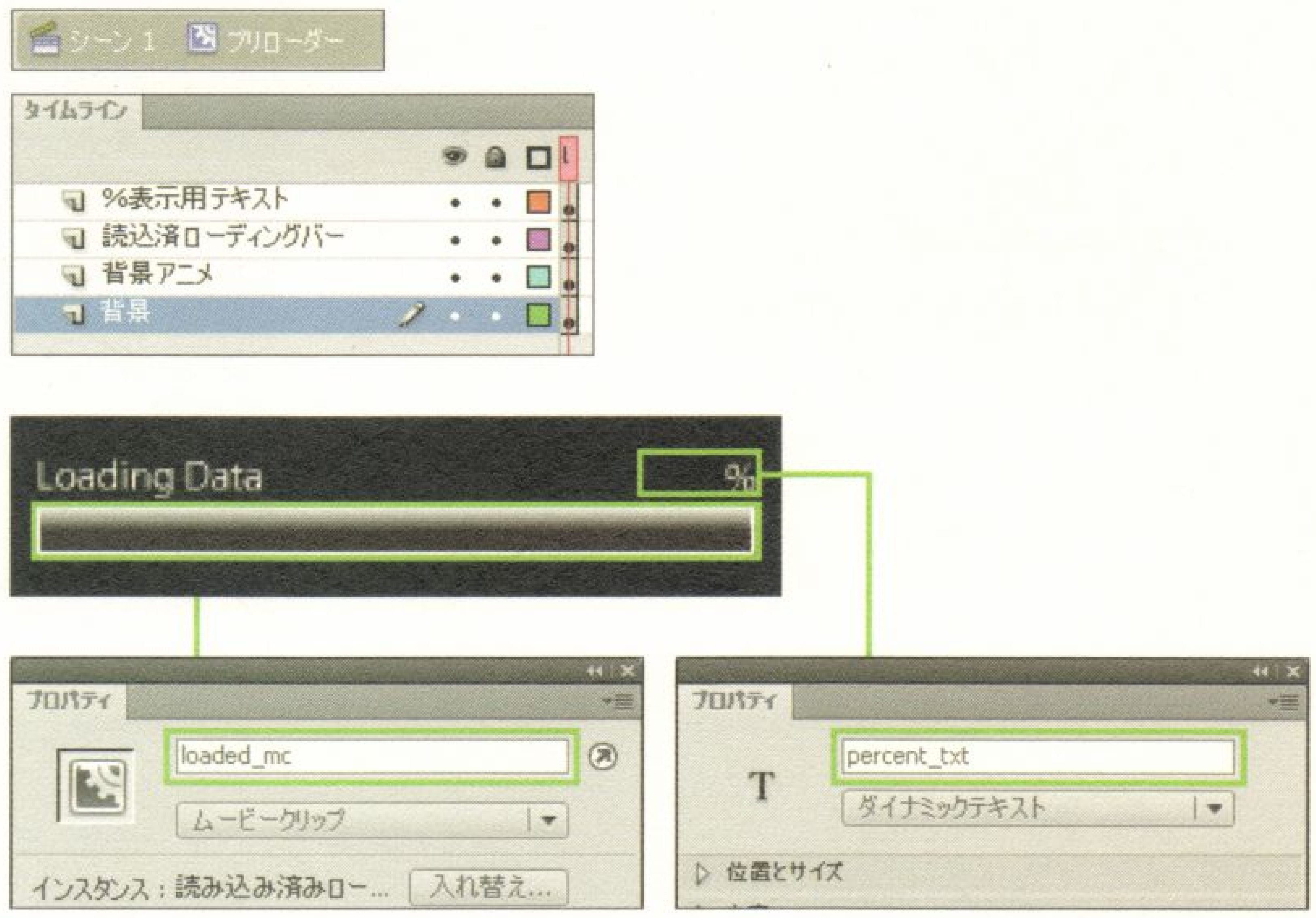
3 コンテンツMovieClip表示用のMovieClipの内部は、各ページごとにタイムラインで分かれています。メニューから、メインタイムライン上のshowContentが呼ばれることで、コンテンツMovieClipの表示が切り替わります。

4 movie.swfのメインタイムラインにムービーが格納されています。このタイムラインをcontents.swfから制御することで、ムービーファイルを外部化することが可能になっています。

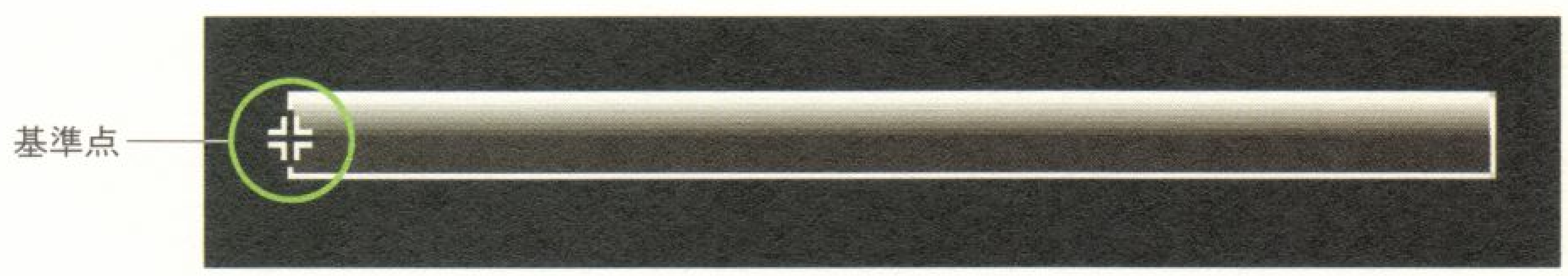
STEP 1 プリローダーを作成する(1)

まずはプリローダーの素材となるムービークリップを作成します。今回のプリローダーには、大きく分けて「読み込み済み分の%表示用テキスト」、「読み込み済み分の表示用のローディングバー」、「背景」の3つの要素があります。

1 「プリローダー」ムービークリップを作成していきます。「読み込み済み分の%表示用テキスト」にはダイナミックテキストのテキストボックスで「percent_txt」、「読み込み済み分の表示用のローディングバー」には「読み込み済みローディングバー」ムービークリップを配置し、「loaded_mc」とインスタンス名を設定します。

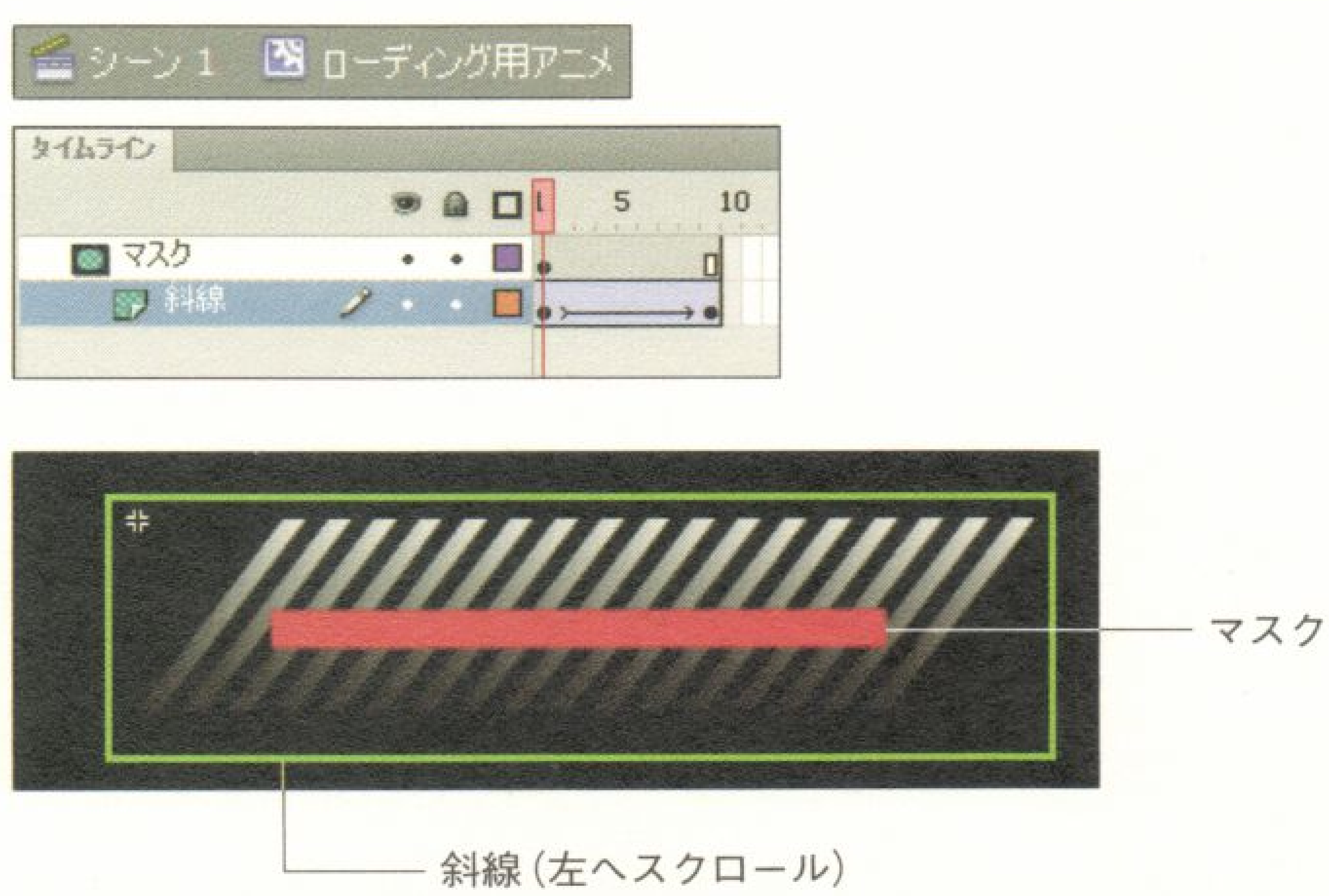


2 「loaded_mc」は、背景を覆い隠した100%読み込みが完了した状態で作成しておきます。これは、後にプリローダーのActionScript部分を実装する際に、読み込み状況の割合と「loaded_mc」の横の大きさの割合を連動させるためです。また今回は、ローディングバーが左から右に伸びていくアニメーションを想定しているため、ムービークリップの左端に基準点を設定しています。



3 ローディングバーの背景用にループアニメーションを作成します。
ここでは、斜線が横にスクロールするタイムラインアニメーションに、背景の大きさに作成したマスクを適用させることで、ループアニメーションを作成しています。

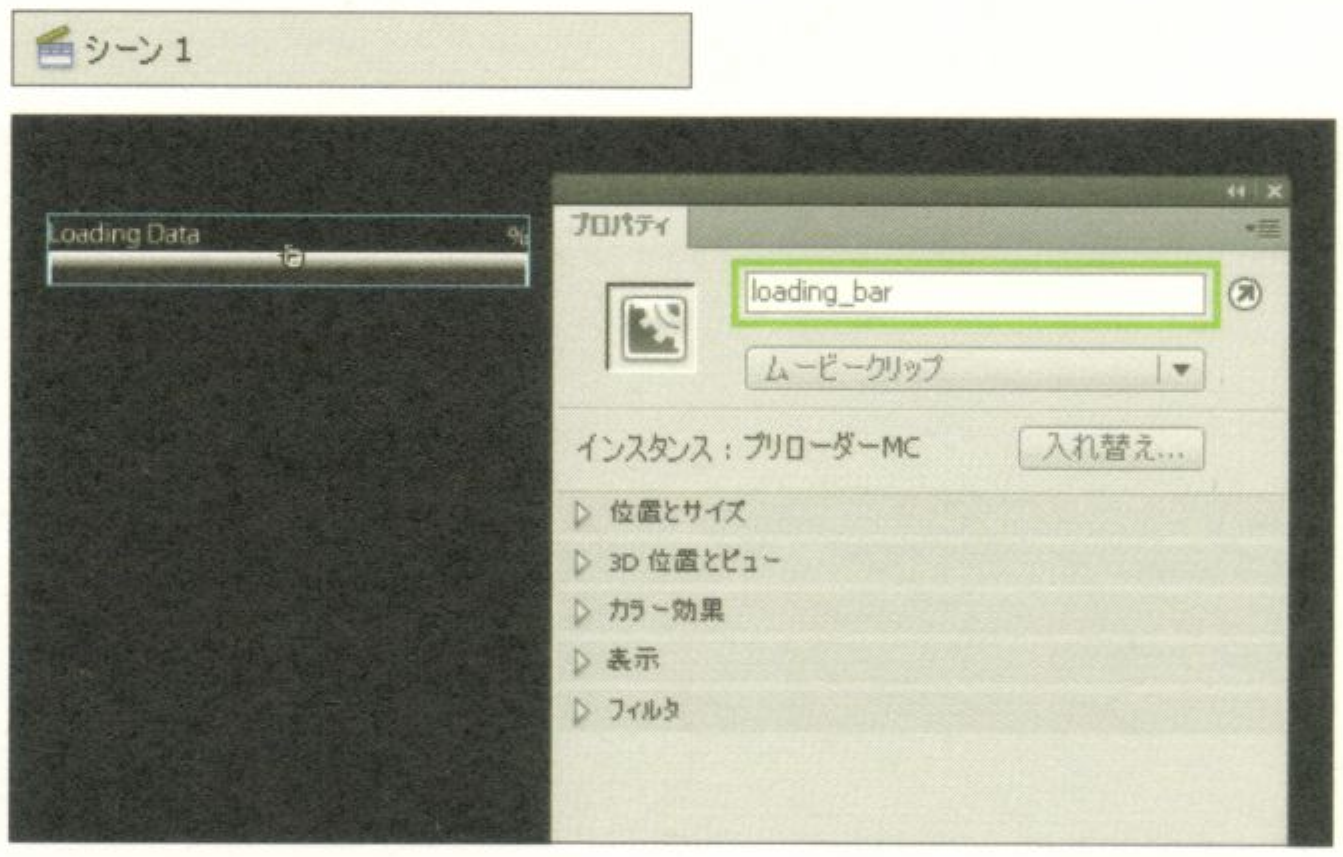
■ マスクの使い方 …… P052へ



STEP 2 プリローダーを作成する(2)

STEP 1で作成したプリローダーの素材を使って、
contents.swfの読み込み状況を表示するプリローダーを作成します。

1 メインタイムラインに先ほど作成したプリローダーを配置し、インスタンス名を「loading_bar」とします。



2 SWFの読み込み状況を取得するための変数「swf_loaderInfo」を作成し、メインタイムラインのプロパティ「loaderInfo」を格納します。この「loaderInfo」はLoaderクラスのプロパティ「contentLoaderInfo」と同様に、読み込み状況の通知・確認や、読み込んだコンテンツの取得が行うことができます。
swf_loaderInfoにはaddEventListenerで「ProgressEvent.PROGRESS」「Event.COMPLETE」を登録し、読み込み状況の進捗と読み込み完了の通知を受け取れるようにします。

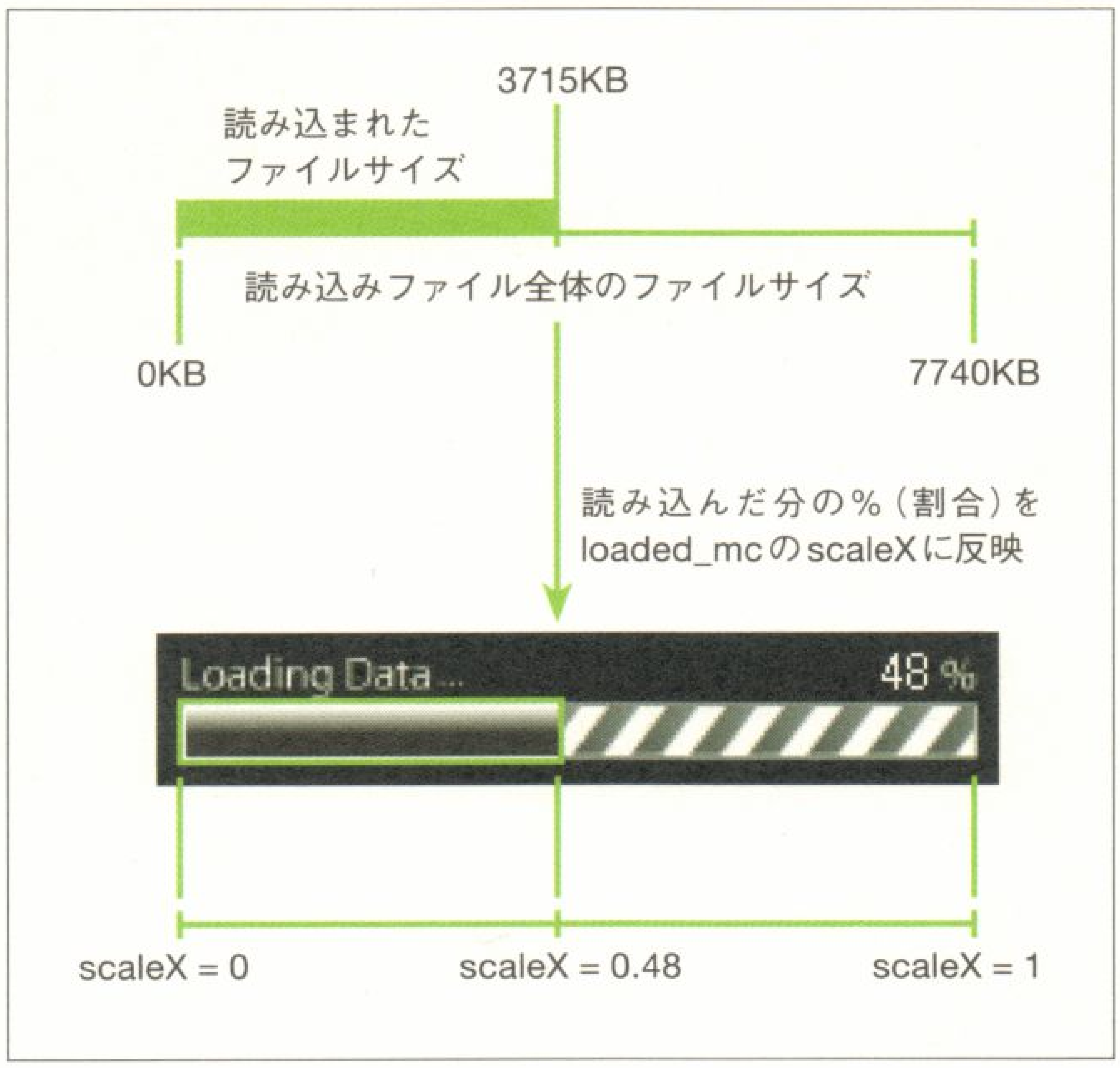
■ Loader ………………………… P158へ

メインタイムライン1フレーム目

```
var swf_loaderInfo = this.loaderInfo;
swf_loaderInfo.addEventListener(ProgressEvent.PROGRESS, onProgress);
swf_loaderInfo.addEventListener(Event.COMPLETE, onLoaded);
```

3 SWFの読み込み状況をローディングバーに表示します。最新の読み込み状況の取得と反映を「function onProgress」で行っています。
読み込んだSWFの%は「(読み込み済みの容量/SWFの総容量)×100」で求めることができます。求めた値は四捨五入してプリローダー内のpercent_txt.textに代入します。また、求めた値を100で割った値をプリローダー内のloaded_mc.scaleXに代入することで、現在の読み込みの割合をローディングバーの長さに適用しています。

■ テキストフィールド……………………… P134へ



メインタイムライン1フレーム目

```
function onProgress(event){
    var percent = Math.floor(swf_loaderInfo.bytesLoaded / swf_loaderInfo.bytesTotal * 100);
    loading_bar.percent_txt.text = String(percent);        ⬆ 読み込んだ割合を%で計算して四捨五入
    loading_bar.loaded_mc.scaleX  = percent/100;
};
```

---

4 SWFの読み込みが完了した際の処理を「function onLoaded」で行います。swf_loadedInfoから「ProgressEvent.PROGRESS」「Event.COMPLETE」に対してのリスナーを削除した後、メインタイムラインを再生します。

■ イベントの概念 ……………………… P077へ

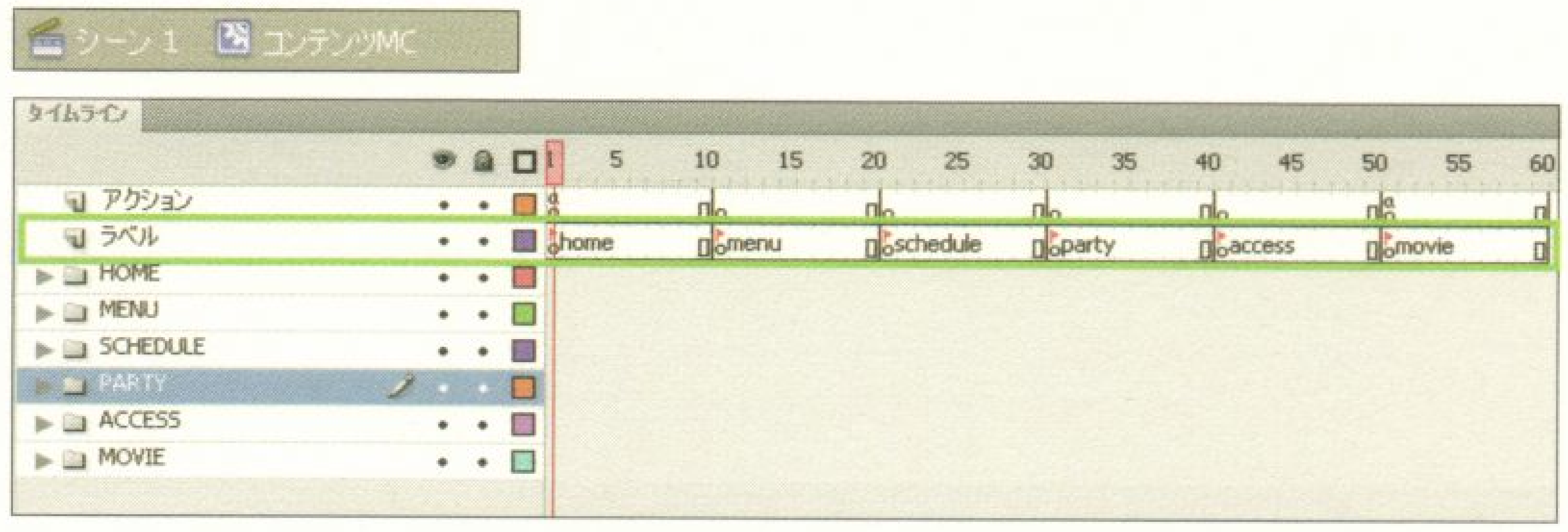
メインタイムライン1フレーム目

```
function onLoaded(event) {
    swf_loaderInfo.removeEventListener(ProgressEvent.PROGRESS, onProgress);
    swf_loaderInfo.removeEventListener(Event.COMPLETE, onLoaded);
    this.play();
};
```

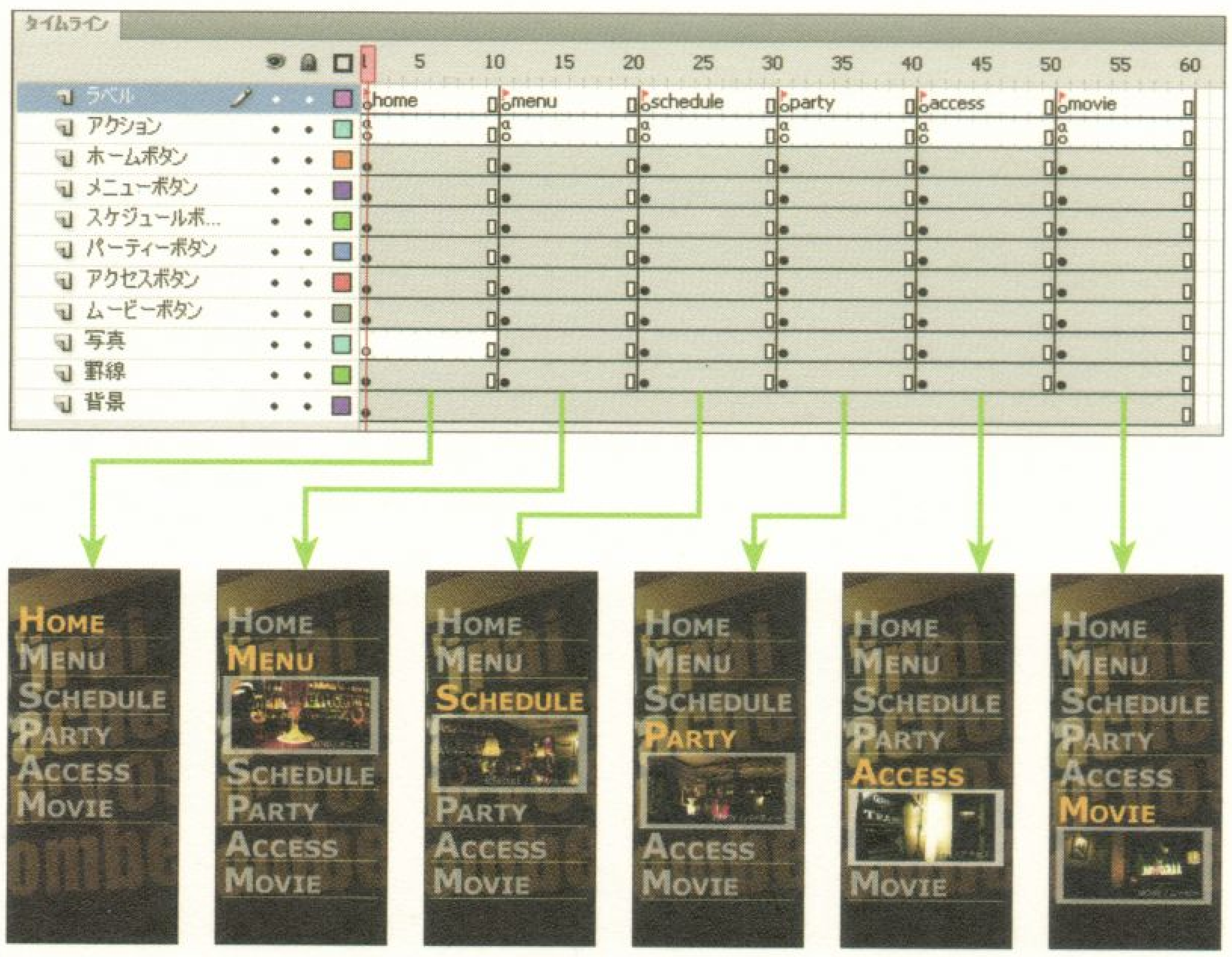
STEP 3 各ページでのメニュー・コンテンツMCの作成

各ページで表示するためのメニュー・コンテンツをそれぞれ作成していきます。今回はメニューとコンテンツとの連携が行いやすいようにタイムライン上のフレームラベルを使ってページを管理しています。

1 各ページの内容をタイムラインで分けたムービークリップ「コンテンツMC」を作成します。
それぞれのフレームラベルには、ページ名に合わせてそれぞれ「home」「menu」「schedule」「party」「access」「movie」が設定されています。



2 各ページでのメニューの表示状態をタイムラインで分けたムービークリップ「メニューMC」を作成します。こちらでもコンテンツMCのムービークリップと同様に、それぞれのページ名に合わせてフレームラベルを設定しています。



3 各ページへの遷移を管理するために「function showContent」をメインタイムラインに作成します。
ここでは、引数に遷移したいフレームラベルを指定することで、メニューMCとコンテンツMCの両方の表示を切り替えるようにしています。

■引数 ………………………………………… P088へ

シーン 1

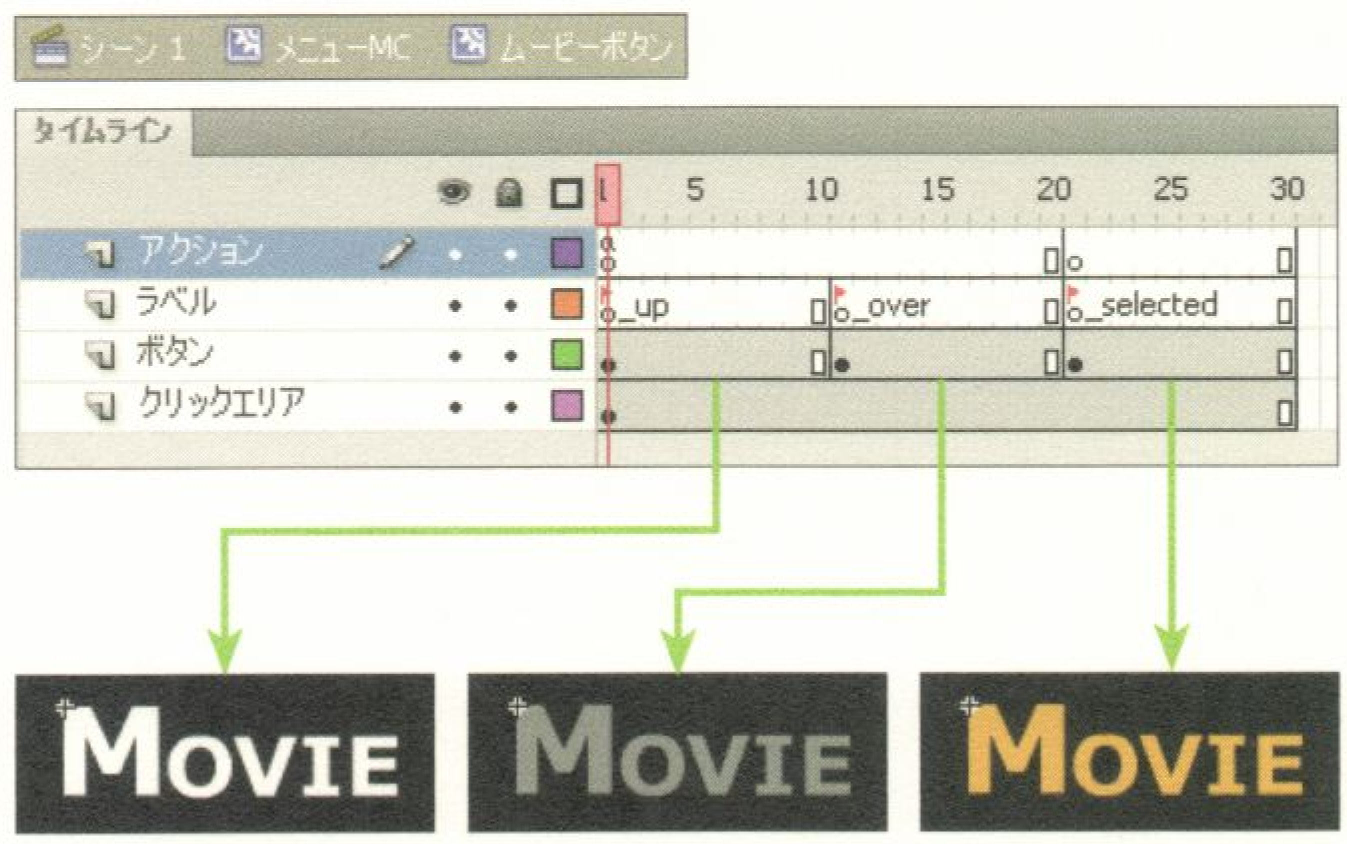
メインタイムライン1フレーム目

```
function showContent(frame_label){
    menu_mc.gotoAndStop(frame_label);
    content_mc.gotoAndStop(frame_label);
};
```

STEP 4 ボタンで遷移できるようにする

STEP 3で作成したメニュー内のボタンから各ページへの遷移を可能にし、
それぞれのページ上でボタンのON/OFFの切り替えを行えるようにします。

1 メニューMC内のそれぞれのボタンを「通常時」「ロールオーバー時」「選択時」の3パターンの表示に分けるために、それぞれを「_up」「_over」「_selected」のフレームラベルが設定されたキーフレームに分けます。今回のサンプルでは「_up」「_over」はbuttonModeがtrueの際に自動的に切り替えられ、「_selected」はページ変更時にプログラム上で切り替えています。



2 各ボタンにクリック時の処理を追加します。すべてのボタンにMouseEvent.CLICKのリスナーを登録するための、メニューMCの1フレーム目に初期化用のキーフレームを追加します。
各ボタンに対し、MouseEvent.CLICKのリスナーを追加します。それぞれのコールバック関数には、メインタイムライン上のfunction showContentを呼んで、クリックしたボタンに対応するページへ遷移する処理を記述しています。また、単純にparent.showContent( )では呼び出せないため、MovieClip(parent).showContent( )として呼び出しています。
MovieClip( )の引数に表示要素を指定すると、その表示要素がMovieClipに変換されるので、自分で追加した変数や関数の呼び出しが可能になります。



```
addEventListenerToBtns();
gotoAndStop("home");
  ⬆ 各ボタンへの設定が終わった後、フレームラベル"home"へ移動
function addEventListenerToBtns(){
  home_btn.addEventListener(MouseEvent.CLICK,onClickHomeBtn);
  menu_btn.addEventListener(MouseEvent.CLICK,onClickMenuBtn);
  schedule_btn.addEventListener(MouseEvent.CLICK,onClickScheduleBtn);
  party_btn.addEventListener(MouseEvent.CLICK,onClickPartyBtn);
  access_btn.addEventListener(MouseEvent.CLICK,onClickAccessBtn);
  movie_btn.addEventListener(MouseEvent.CLICK,onClickMovieBtn);
}

function onClickHomeBtn(event){
  MovieClip(parent).showContent("home");
}
function onClickMenuBtn(event){
  MovieClip(parent).showContent("menu");
}
function onClickScheduleBtn(event){
  MovieClip(parent).showContent("schedule");
}
function onClickPartyBtn(event){
  MovieClip(parent).showContent("party");
}
function onClickAccessBtn(event){
  MovieClip(parent).showContent("access");
}
function onClickMovieBtn(event){
  MovieClip(parent).showContent("movie");
}
```

3 メニュー内で各ページに表示が切り替わった際に、表示中のページのボタンをクリック不可になるようにします。各ページのフレームではまず、すべてのボタンを有効にするfunction initBtns( )を呼び、その後、表示中のページのボタンの機能を無効にしています。

メニューMC内movieフレーム

```
//メニューMCがMOVIEの表示になる場合
initBtns();
movie_btn.buttonMode = false;
movie_btn.mouseEnabled = false;
movie_btn.gotoAndStop("_selected");
```

メニューMC内1フレーム目

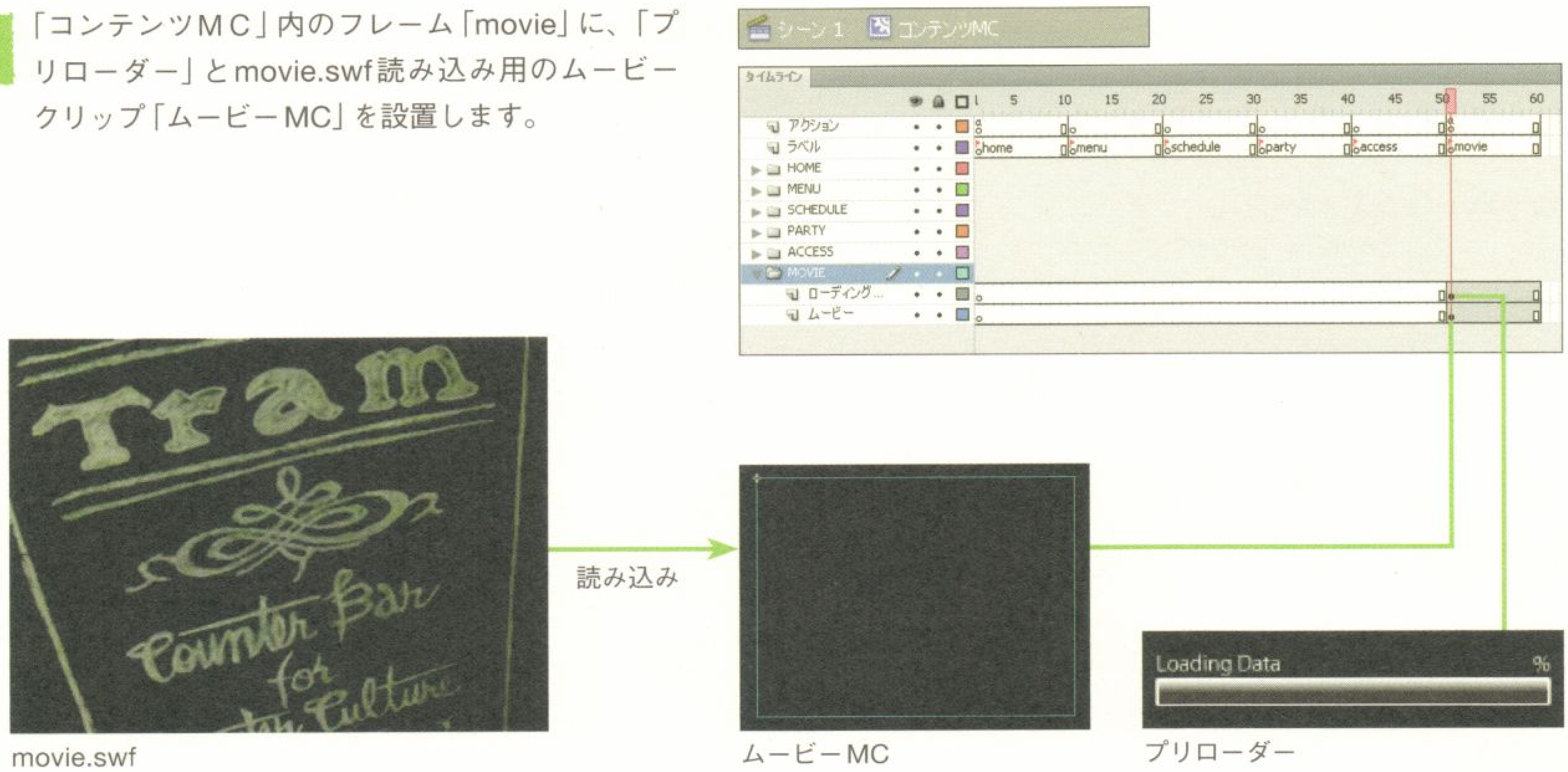
```
function initBtns(){
  initBtn(home_btn);
  initBtn(menu_btn);
  initBtn(schedule_btn);
  initBtn(party_btn);
  initBtn(access_btn);
  initBtn(movie_btn);
}

function initBtn(target_btn){
  target_btn.gotoAndStop("_up");
  target_btn.buttonMode = true;
  target_btn.mouseEnabled = true;
  target_btn.mouseChildren = false;
}
```

STEP 5 外部SWFを読み込み、制御する

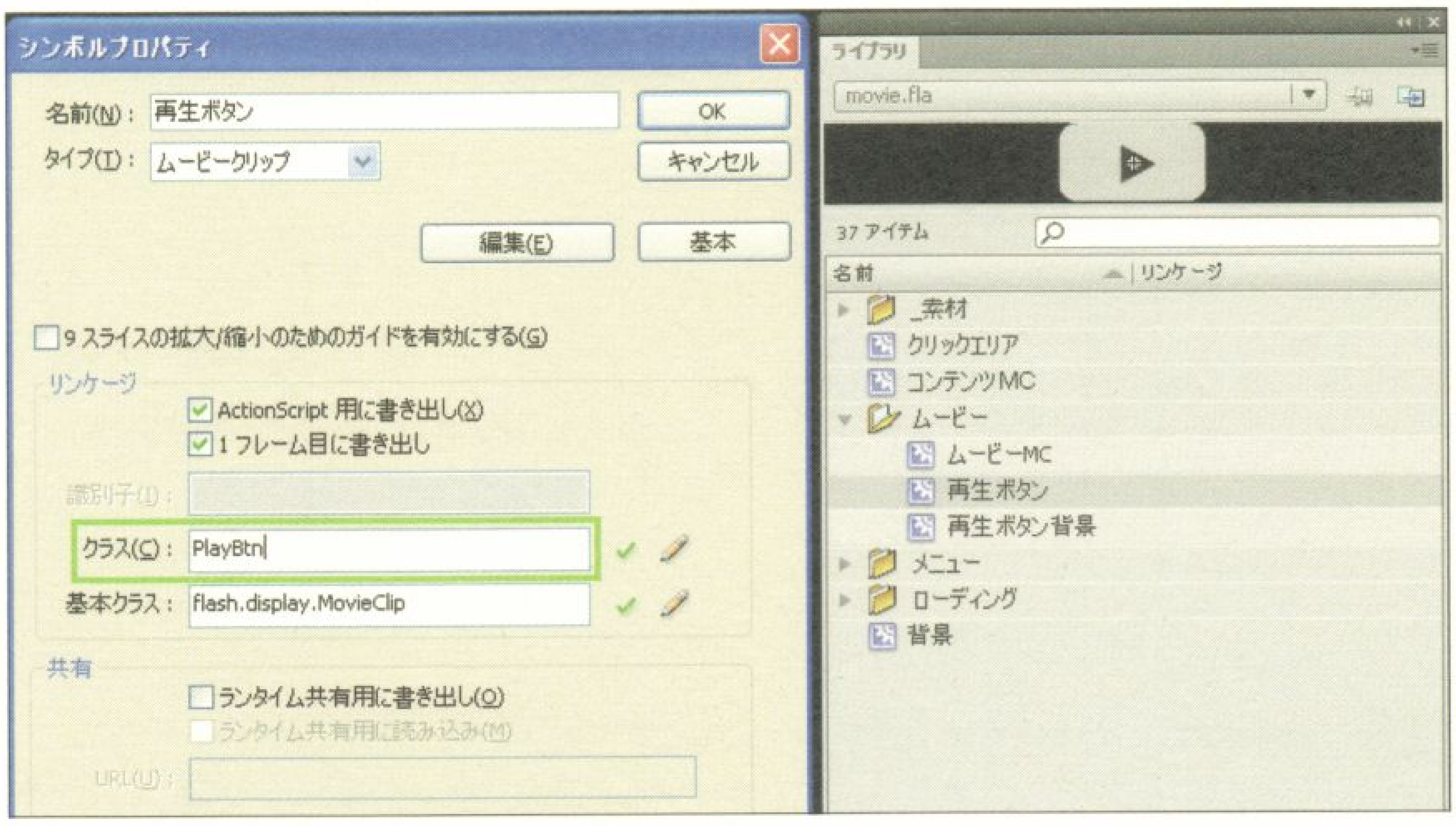
MOVIEページ内で、外部SWFを読み込み、再生を制御する仕組みを作成します。Loaderを使って読み込んだSWFは、通常のムービークリップと同様にタイムラインや位置の制御などを行うことが可能です。

1 「コンテンツMC」内のフレーム「movie」に、「プリローダー」とmovie.swf読み込み用のムービークリップ「ムービーMC」を設置します。



---

2 動的に配置する「再生ボタン」のリンケージ設定を行います。この設定を行うことで、「再生ボタン」を動的に設定することが可能です。ここでは、クラス名を「PlayBtn」としました。



---

3 読み込んだムービークリップを格納するための変数「swf_movie」と、動的に配置する再生ボタンを格納するための変数「play_btn」を、それぞれ宣言しておきます。

コンテンツMC　フレーム「movie」

```
var swf_movie;
var play_btn;
```

PRACTICE 7

4 Loaderクラスを使って「movie.swf」を読み込みます。
リスナーの登録の対象が、this.loaderInfoかloader.contentLoaderInfoかという違いはありますが、基本的にはSTEP 1で解説したプリローダーと同様です。

コンテンツMC　フレーム「movie」

```
var loader = new Loader();
var url_req = new URLRequest("movie.swf");
loader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, onProgress);
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoaded);
loader.load(url_req);

function onProgress(event){
   var percent = Math.floor(loader.contentLoaderInfo.bytesLoaded / loader.contentLoaderInfo.bytesTotal * 100);
   loadingBar.percent_txt.text = String(percent);
   loadingBar.loaded_mc.scaleX  = percent/100;
};
```

movie.swfの読み込み

読み込み中のローディングバーの処理

■ プリローダーを作成する……………… P161へ

5 読み込み終わったmovie.swfをloader.contentLoaderInfo.contentから取得し、変数swf_movieに代入します。そのままの状態では、自動的にswf_movieが再生されてしまうので、タイムラインを停止した後にmovie_mc内に配置します。配置したswf_movieの上には「再生ボタン」を動的に配置します。配置した「再生ボタン」は変数「play_btn」に格納し、MouseEvent.CLICKのリスナー登録をしておきます。

```
function onLoaded(event) {
      loader.contentLoaderInfo.removeEventListener(ProgressEvent.PROGRESS, onProgress);
      loader.contentLoaderInfo.removeEventListener(Event.COMPLETE, onLoaded);
      swf_movie = loader.contentLoaderInfo.content;
      swf_movie.x = 10;
      swf_movie.y = 10;
      swf_movie.stop();
      movie_mc.addChild(swf_movie);

      //ローディングバーを消去
      removeChild(loadingBar);

      play_btn = new PlayBtn();
      play_btn.x = movie_mc.width/2;
      play_btn.y = movie_mc.height/2;
      play_btn.buttonMode = true;
      play_btn.addEventListener(MouseEvent.CLICK,onClickPlayBtn);
      movie_mc.addChild(play_btn);
};
```

読み込んだswfを画面に配置

再生ボタンの配置

6 play_btnをクリックして実行される関数「onClickPlayBtn」を作成します。
play_btnがクリックされると、play_btnのリスナーを削除し、画面上からplay_btnを削除します。play_btnが見えなくなった後、swf_movieが再生されます。

```
function onClickPlayBtn(event:MouseEvent):void{
   play_btn.removeEventListener(MouseEvent.CLICK,onClickPlayBtn);
   movie_mc.removeChild(play_btn); ← 再生ボタンを画面から削除

   swf_movie.play(); ← 読み込んだswfの再生
}
```

STEP 6 メニューの位置をステージサイズに合わせ調整する

ステージサイズが変更された際に、
メニューの位置を自動で調整するようにします。

1 今回のデザインでは、左上の隅を基準としてステージサイズが可変するようにします。stage.alignにStageAlign.TOP_LEFTを設定します。

■ ステージの持っている情報…………… P145へ

シーン1

メインタイムライン 1フレーム目

```
stage.align = StageAlign.TOP_LEFT;
```

2 ステージのサイズが変更された際にstageから発行されるEvent.RESIZEにリスナーを登録します。

■ ステージサイズの変化に応じて情報を取得する
………………………………………… P146へ

メインタイムライン 2フレーム目

```
stage.addEventListener(Event.RESIZE,onResizeStage);
```

3 ステージサイズの変更に合わせて、メニューの位置を変更します。ここではメニューの位置を(ステージの横幅 - 270px)に、メニューの背景の高さをステージの縦幅と等しくなるように設定しています。
また、ステージの横幅が狭くなりすぎた場合、メニューがコンテンツ部分に被ってしまうので、メニューの位置が400pxより左側に来ないようにif文を用いています。

メインタイムライン 2フレーム目

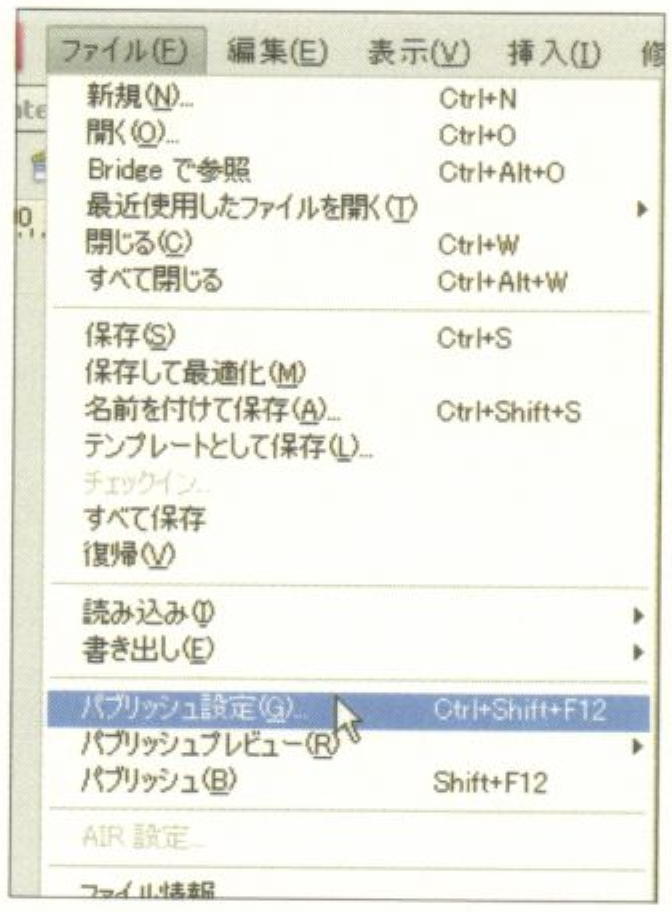
```
function onResizeStage(event){
    var menu_pos = stage.stageWidth - 270;
    //メニューが左端に行き過ぎないよう制限をかける
    if(menu_pos < 400){
        menu_pos = 400;
    }
    menu_mc.x = menu_pos;
    menu_mc.background_mc.height = stage.stageHeight;
}
```



STEP 7 パブリッシュ設定を行う

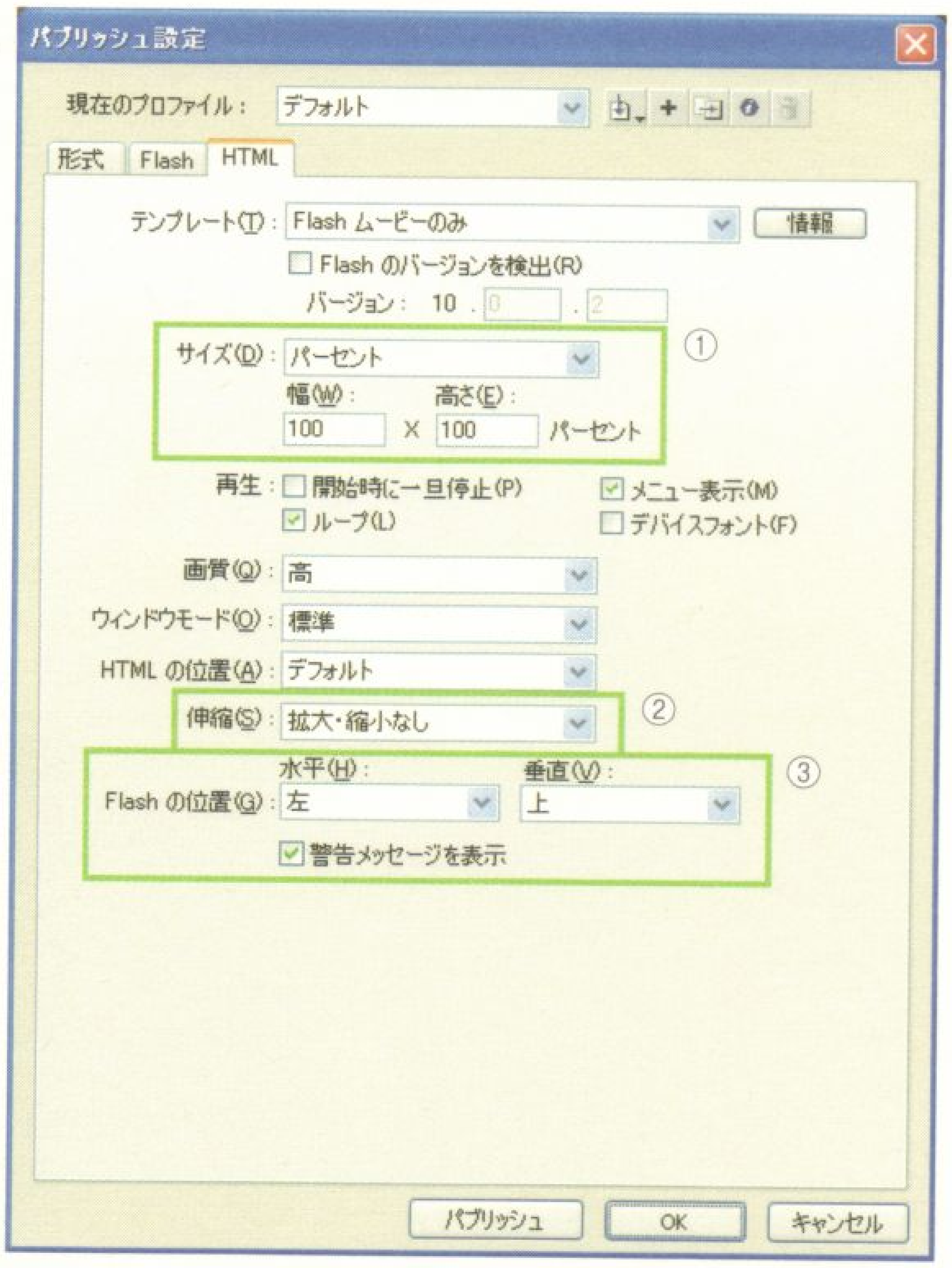
デフォルトの設定ではパブリッシュしたHTMLで、
SWFコンテンツのサイズが可変にならないのでパブリッシュ設定を行います。

1 ［ファイル→パブリッシュ設定］からパブリッシュ設定を開きます。



2 HTMLタブを選択し、「サイズ」「伸縮」「FLashの位置」の設定を行います。
今回は左上の隅を基準にして拡大・縮小なしでサイズを可変させたいので、それぞれ下記のように設定します。

①「サイズ」を「パーセント」にして、幅と高さをそれぞれ100に設定
②「伸縮」を「拡大・縮小なし」に設定
③「Flashの位置」の「水平」を「左」に、「垂直」を「上」にそれぞれ設定



3 この状態で「パブリッシュ」ボタンを押すと、サイズ可変に対応したHTMLが書き出されます。
また、この設定を保存するには「OK」ボタンを押します。

応用

Ex.1 ローディング完了からの画面遷移をスムーズにする

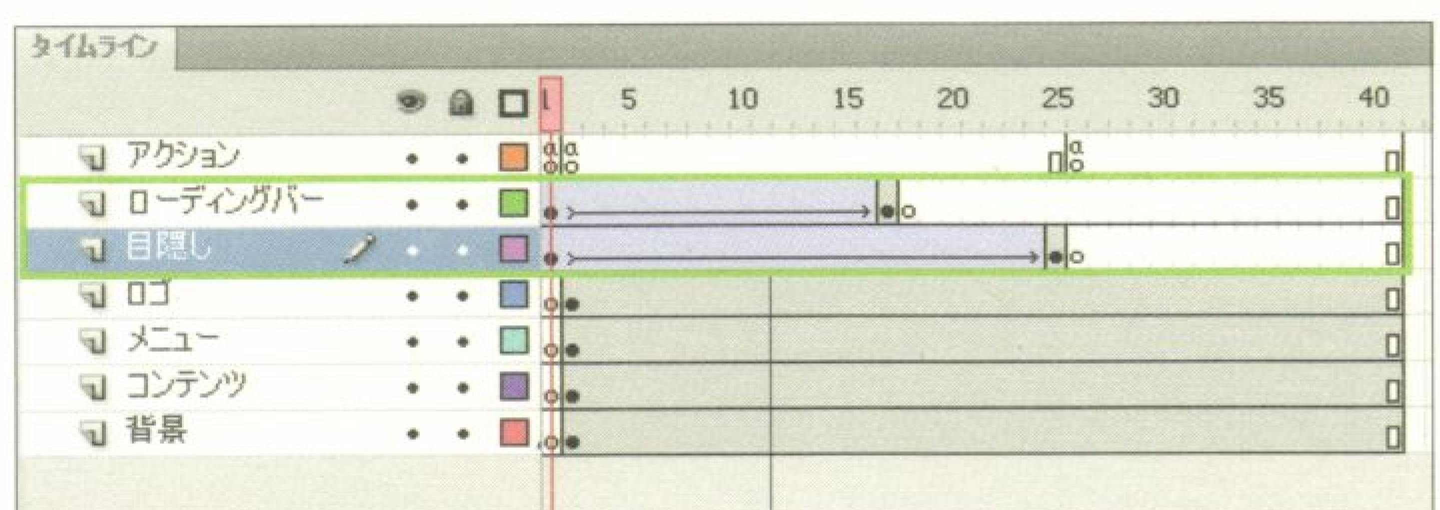
現状では読み込みが終わると「パッ」と画面が切り替わってしまうため、
自然なフェードインになるようにメインタイムラインを調節します。



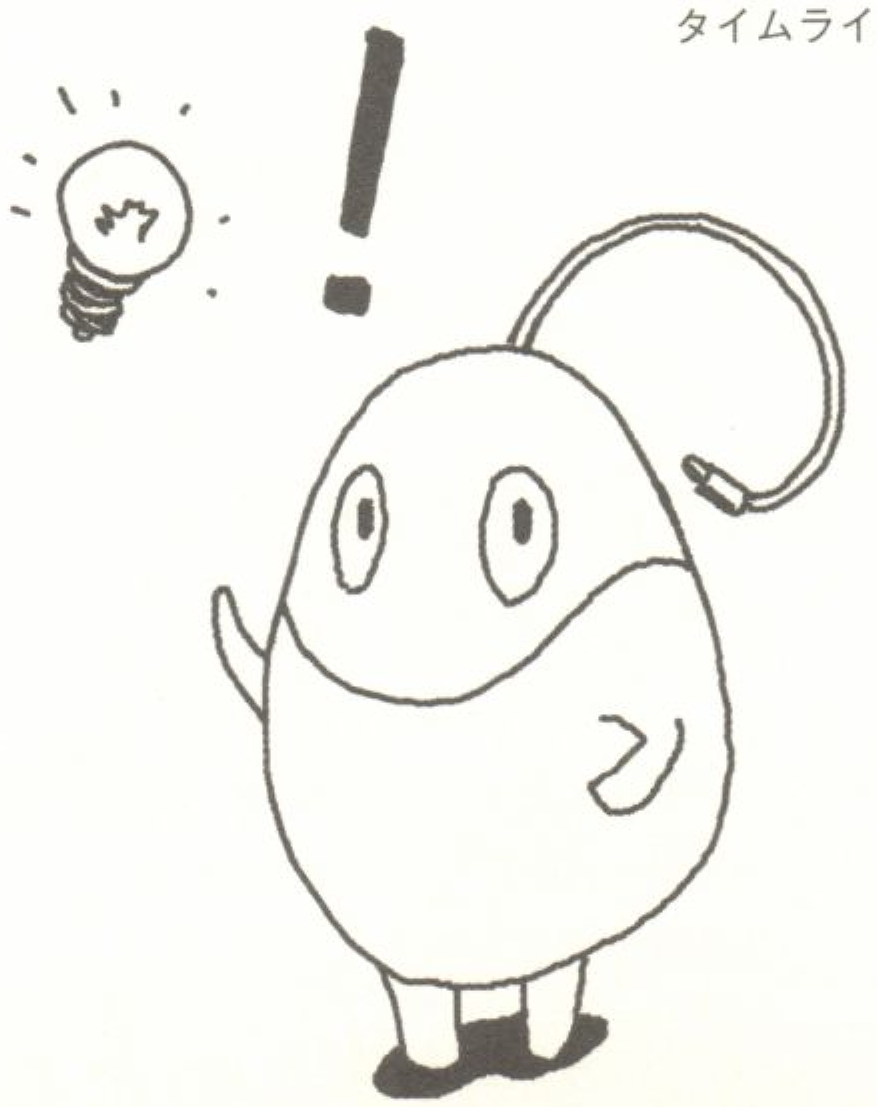
1 ローディングが完了した後、背景やメニューがフェードインしているように見せるため、ローディングと2フレーム目以降の画面要素の間に黒いムービークリップを設置します。



2 各レイヤーのフレーム数を伸ばし、フェードインのアニメーションを入れるため、時間を確保します。
プリローダーと間に挟んだ黒いムービークリップを、タイムライン上のトゥイーンでアルファを薄くしていきます。
こうすることで、ローディング完了後フェードインでサイトが浮かび上がってくる、自然な演出が実現可能になります。



徐々にアルファが小さくなるアニメーションを
タイムライン上で作成



PRACTICE 7

PRACTICE 8

ウェブカメラとキーボードを使ったコンテンツを作る

表現力：★★☆☆☆　プログラミング力：★★★★★

このサンプルでは、「キーボード」、「ウェブカメラ」といったFlashの画面の外にあるデバイスを使ったコンテンツを作ります。外部デバイスをプログラミングで制御する際には、注意すべき点がいくつかあります。ここではそういった点を解説しながら、サンプルを作成していきます。

このサンプルで必要な主な基礎技術

このサンプルを作成するにあたって、以下の知識が必要になります。
もし分からない項目があれば、以下のページで基礎技術を学んでください。

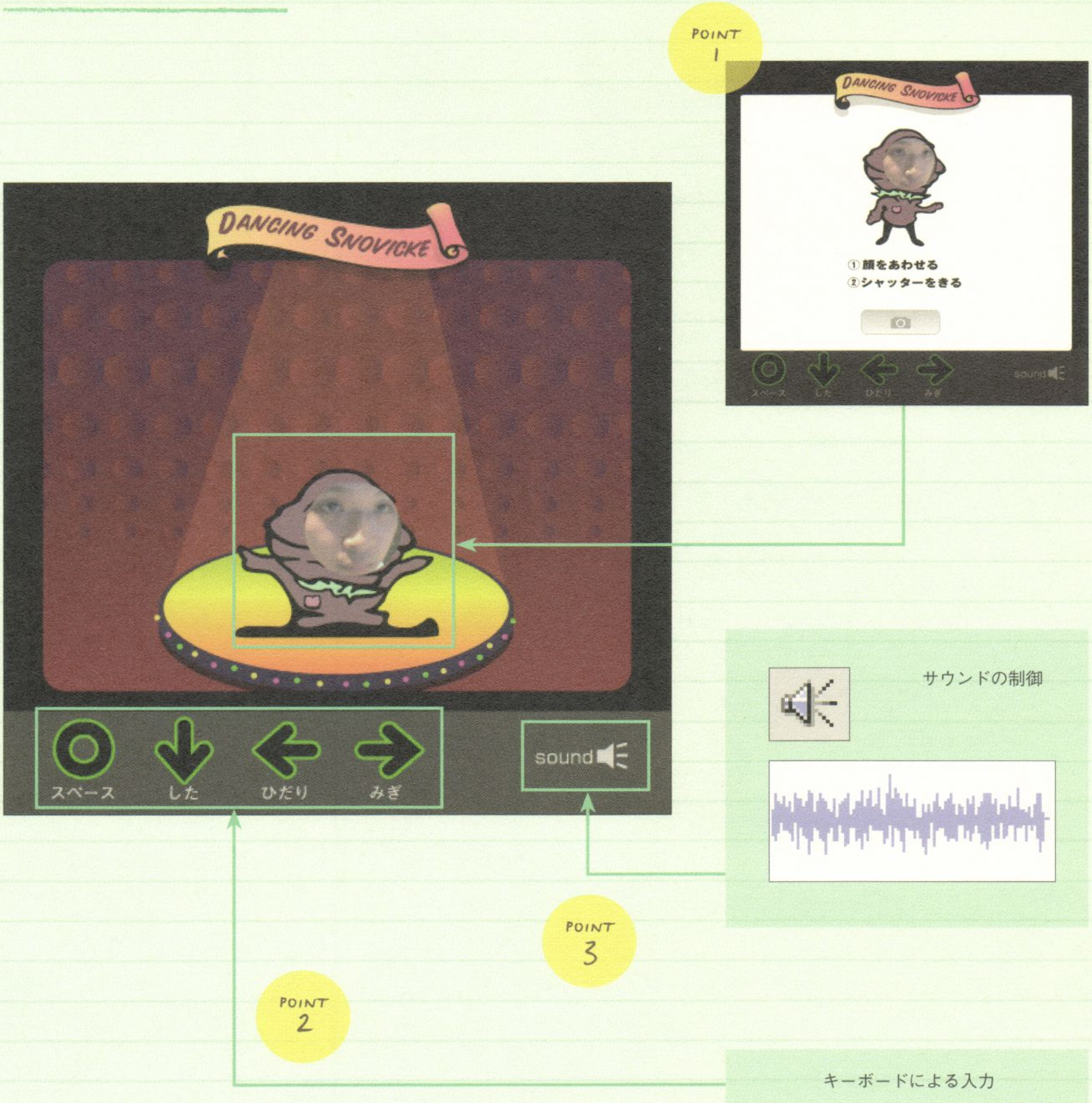
- タイムラインを制御する……P080へ
- MovieClipを制御する……P083へ
- addChild……P124へ
- ActionScriptでのサウンド制御……P151へ
- キーで操作する……P155へ
- ウェブカメラ……P169へ

このサンプルで学ぶ応用技術

このサンプルを通して、以下の応用技術を学びます。
基礎知識だけではカバーできない実制作で必要な知識です。

- ウェブカメラで写した画像のキャプチャ
- キーボードによるコンテンツ操作
- ユーザー操作による音声の制御

サンプルの説明



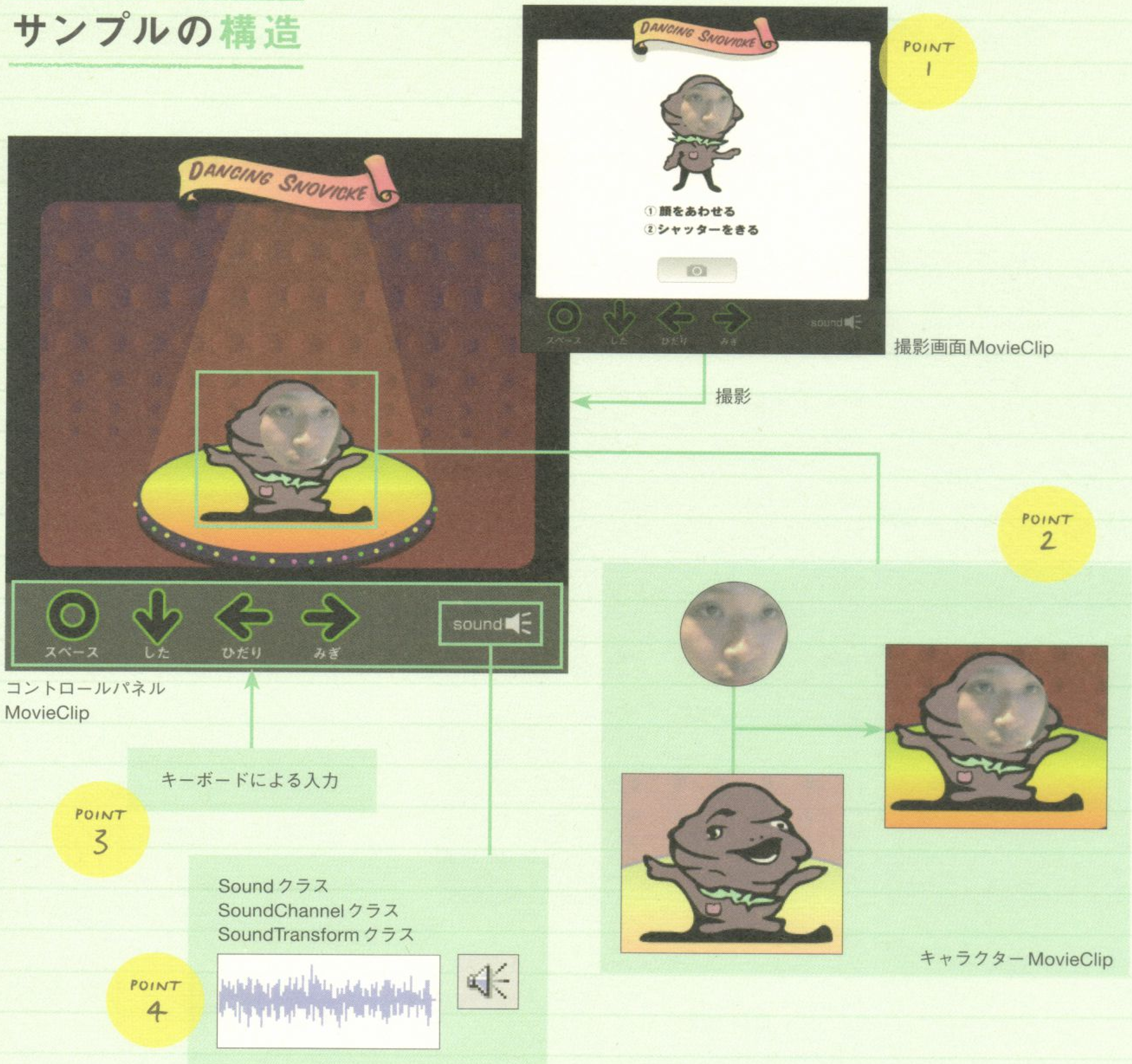
POINT

1 ウェブカメラの制御で自分の顔を撮影し、キャラクターの顔に貼り付けます。

2 キーボード入力によるキャラクターの操作を行います。あらかじめ作成したタイムラインのアニメーションと、プログラムで配置した顔写真との連携が必要になります。

3 プログラムで再生されたBGMのON、OFFを制御します。

サンプルの構造



POINT

1 ウェブカメラで写した映像をリアルタイムで画面に表示します。ボタンを押すとカメラの映像を画像化してキャラクターに貼り付けます。カメラの制御にはCamera クラスを使います。

2 タイムラインによるアニメーションと、プログラムで配置した顔写真との連携を行います。

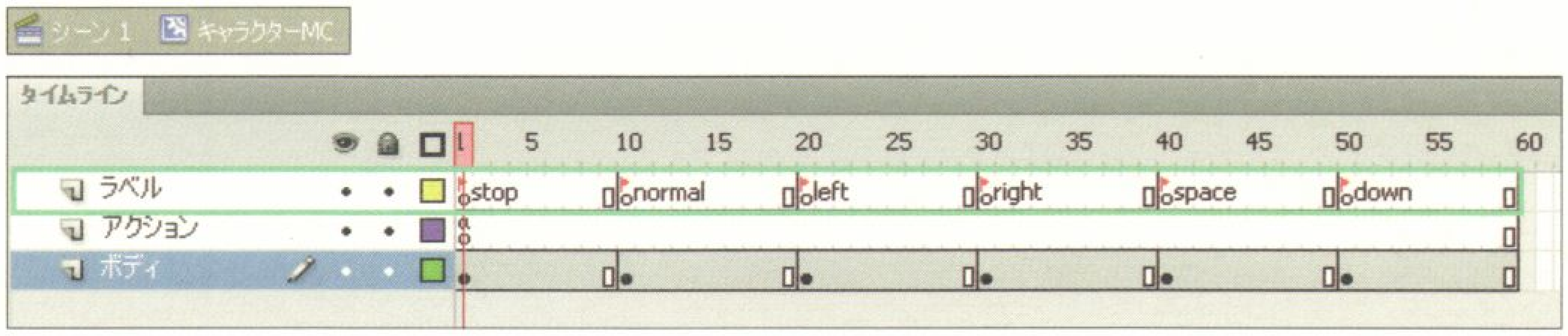
3 Keyboard クラスを利用して、キーボード上で入力されたキーの種類に応じて、キャラクターのアニメーションを操作します。

4 プログラムで再生された音声を制御します。
今回はSound クラス、SoundChannel クラス、SoundTransform クラスを使い、音量のON、OFF をユーザーの操作で行えるようにします。

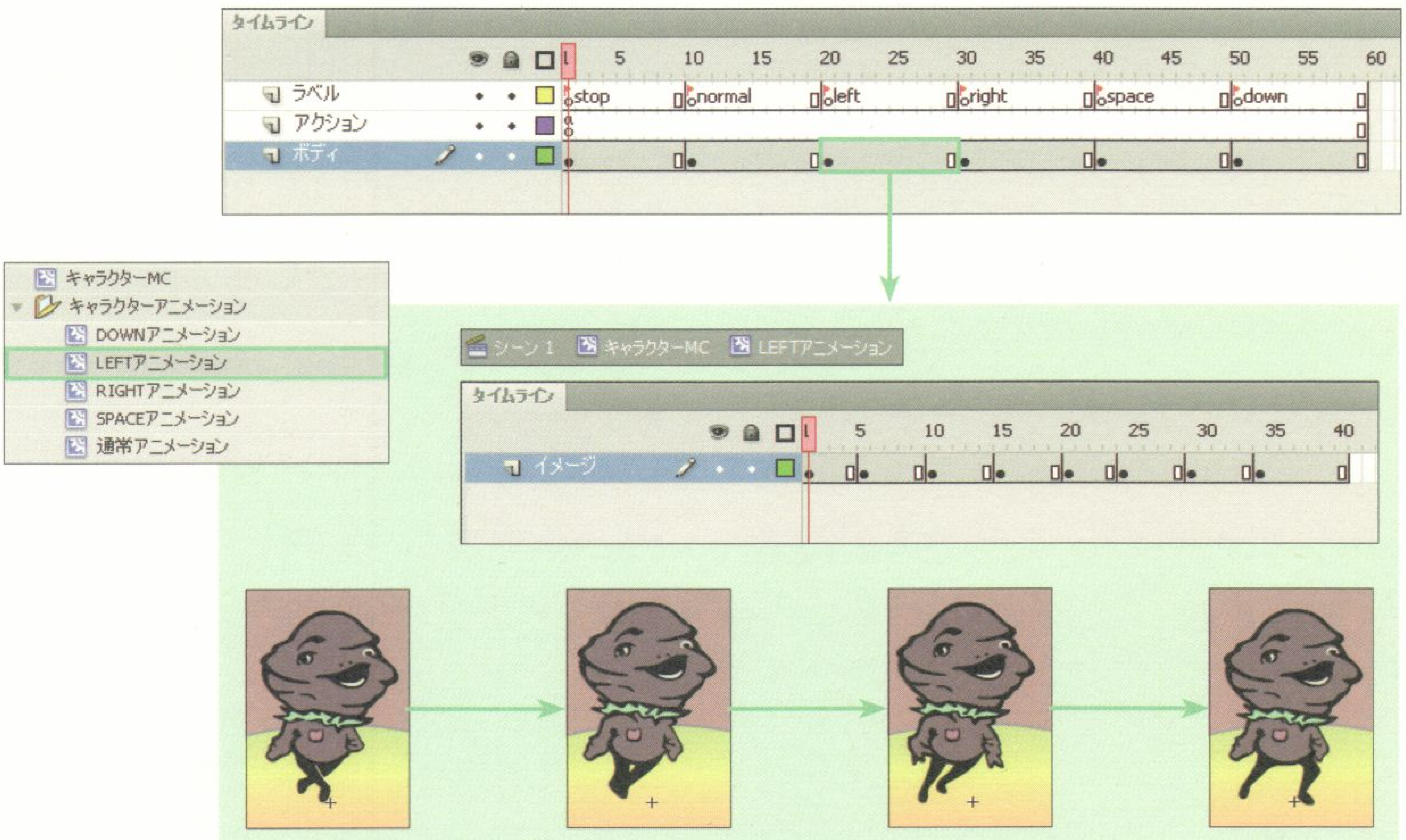
STEP 1 キャラクターのアニメーションの作成

まずはベースとなる、キャラクターのアニメーションを作成します。

1 キャラクターにはキーの操作に応じて、各フレームを用意します。各フレームにはそれぞれフレームラベルを割り当て、キー操作の際に任意のフレームに移動できるようにしておきます。今回は「left」「right」「space」「down」のほかに、通常状態の「normal」と、顔写真撮影時に使用する止まった状態の「stop」を用意しています。



2 各フレームごとにキャラクターをアニメーションさせたムービークリップを作成します。



3 一連のアニメーション終了時には、キャラクターのフレームを「normal」に戻すようにしておきます。

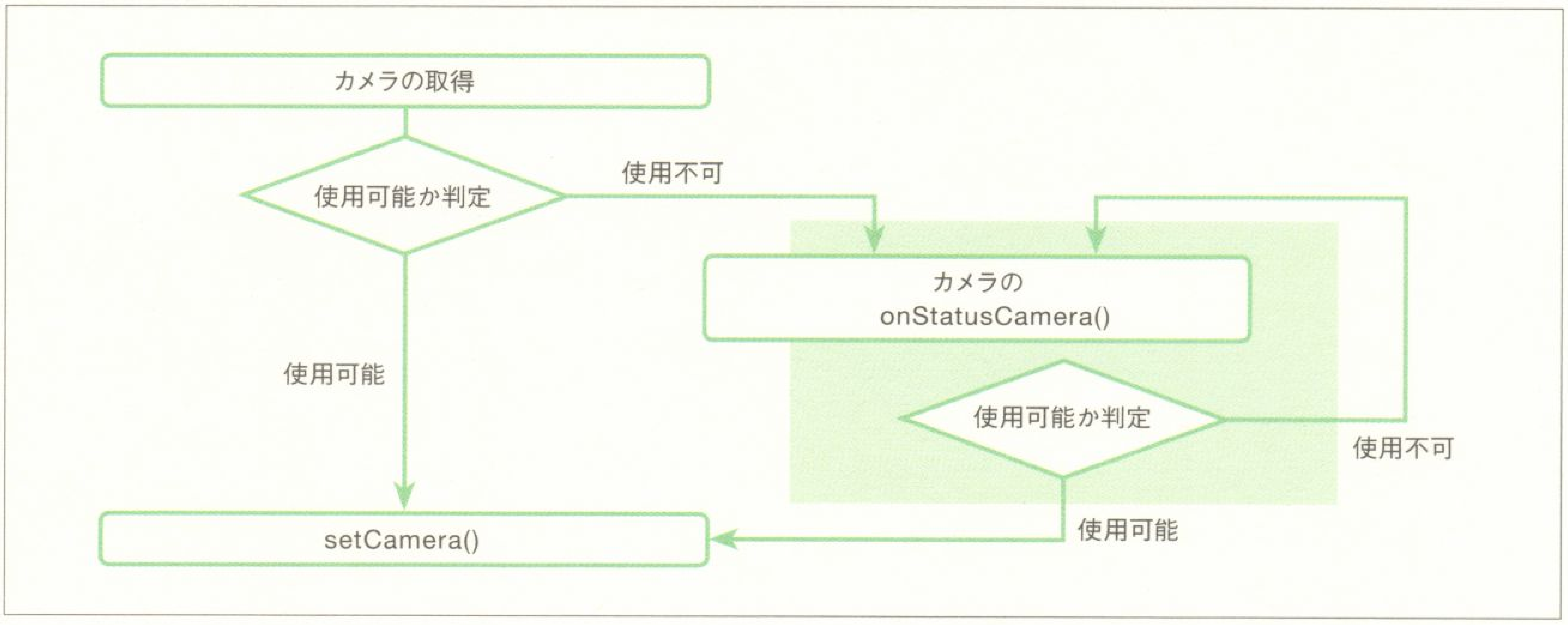
キャラクターの各アニメーションの最後のフレーム

```
MovieClip(parent).gotoAndStop("normal");
```

STEP 2 ウェブカメラの起動と制御

顔写真を撮影するためのウェブカメラを起動し、制御します。

カメラ判定の流れ



■ ウェブカメラ：カメラをActionScriptで制御する ……P169へ

1 カメラの制御に使用するCameraインスタンスを格納するための変数「camera」と、カメラの表示に使用するVideoインスタンスを格納するための変数「video」をそれぞれ宣言します。

シーン1

メインタイムライン 1フレーム目

```
var camera;
var video;
```

2 カメラを取得するメソッド「getCamera」を作成し、呼び出します。
getCamera内では、Camera.getCamera( )で取得したCameraインスタンスを変数cameraに代入し、そのCameraインスタンスが使用可能かどうかを調べています。
Cameraインスタンスのmutedプロパティがtrueな場合は、カメラが使用不可なためSecurity.showSettings(SecurityPanel.PRIVACY);を実行して、［Flash Playerの設定］ダイアログボックスを開きます。また、その際にはCameraインスタンスの使用可能状況の変更の通知を受け取るために、StatusEvent.STATUSに対してリスナーの登録を行っておきます。
カメラが使用可能な場合は、カメラの映像を画面上に表示するメソッド「setCamera( )」を呼びます。

メインタイムライン 1フレーム目

```
function getCamera(){
  camera = Camera.getCamera();

  if(camera.muted == true){ ⬅ カメラが使用可能かどうかの判定
    camera.addEventListener(StatusEvent.STATUS, onStatus);
    Security.showSettings(SecurityPanel.PRIVACY);
    ⬆ ダイアログボックスを開く
  }else{
    setCamera();
  }
}
```

■ if ……………………………… P107へ
■ Camera ……………………………… P169へ

3 メソッド「onStatus」内では、カメラが使用可能になった際はイベントのリスナーを削除し、メソッド「setCamera( )」を呼ぶようになっています。

メインタイムライン 1フレーム目

```
//カメラの使用可能の設定が変更された際の処理

function onStatus(event){
  if (camera.muted == false) {
    camera.removeEventListener(StatusEvent.STATUS, onStatus);
    setCamera();
  }
}
```

4 カメラで撮影した映像は、Videoインスタンス内に「Video.attachCamera( )」メソッドを使って設置した後、Videoインスタンスを配置することで表示します。
また、ボタンをクリックした際にカメラに写った映像を撮影するメソッド「onClickShutterBtn」が呼ばれるように、リスナーを登録しておきます。

```
function setCamera(){
  //カメラの映像を画面に配置
  video = new Video(camera.width, camera.height); ← カメラの映像のサイズのVideoインスタンスを生成
  video.attachCamera(camera); ← カメラの映像をVideoインスタンスに設置
  video.scaleX = -1; ← カメラの位置調整
  video.x = video.width/2; ← カメラの位置調整
  video.y = -video.height/2; ← カメラの位置調整
  camera_mc.face_frame.addChild(video); ← 画面上に映像を配置

  //シャッターボタンの設定
  camera_mc.shutter_btn.buttonMode = true;
  camera_mc.shutter_btn.mouseChildren = false;
  camera_mc.shutter_btn.addEventListener(MouseEvent.CLICK,onClickShutterBtn);
}
```

配置先のムービークリップは、タイムライン上で円形にマスクがかかっていて、カメラに写っている映像が円形に切り取られて表示されるようになっています。

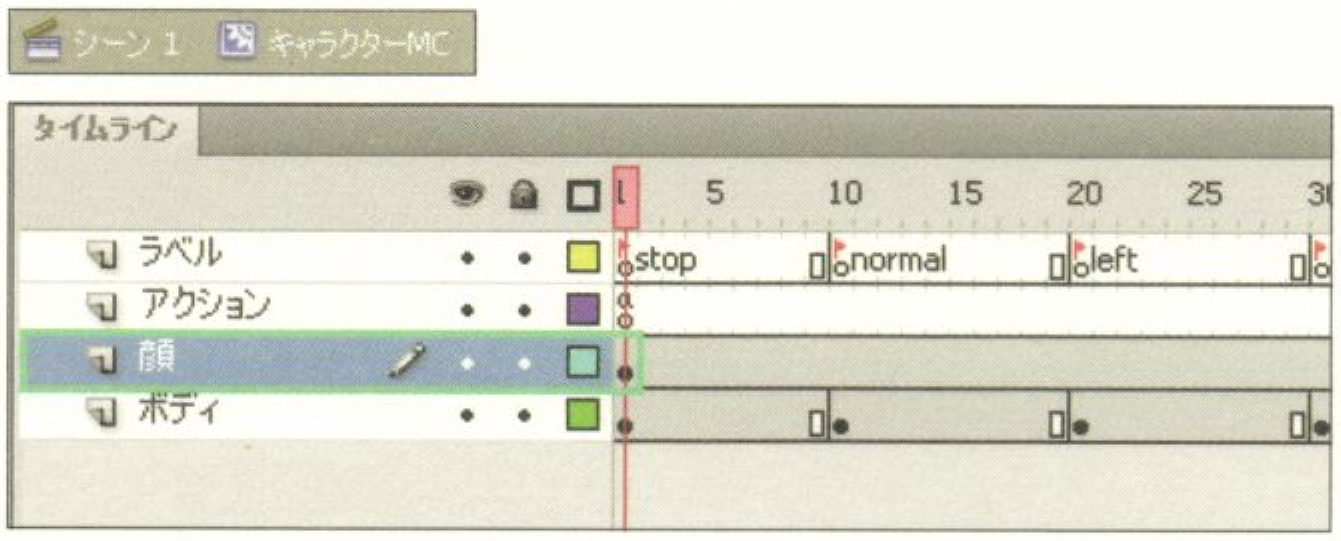
■ マスクを使う ………………………… P051へ

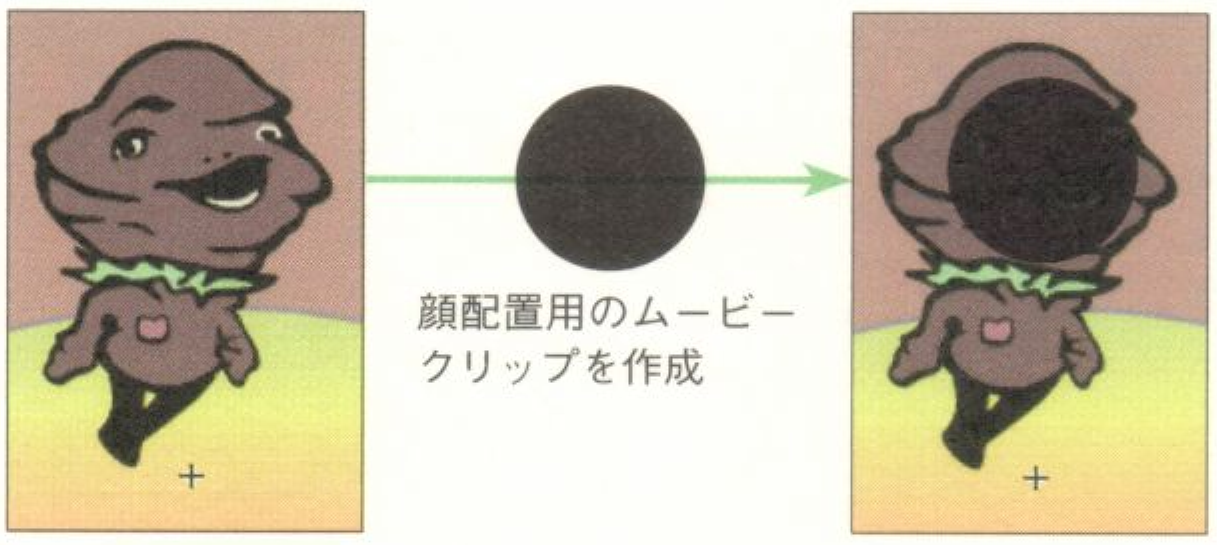


STEP 3 カメラの映像を画像化してキャラクターに貼り付ける

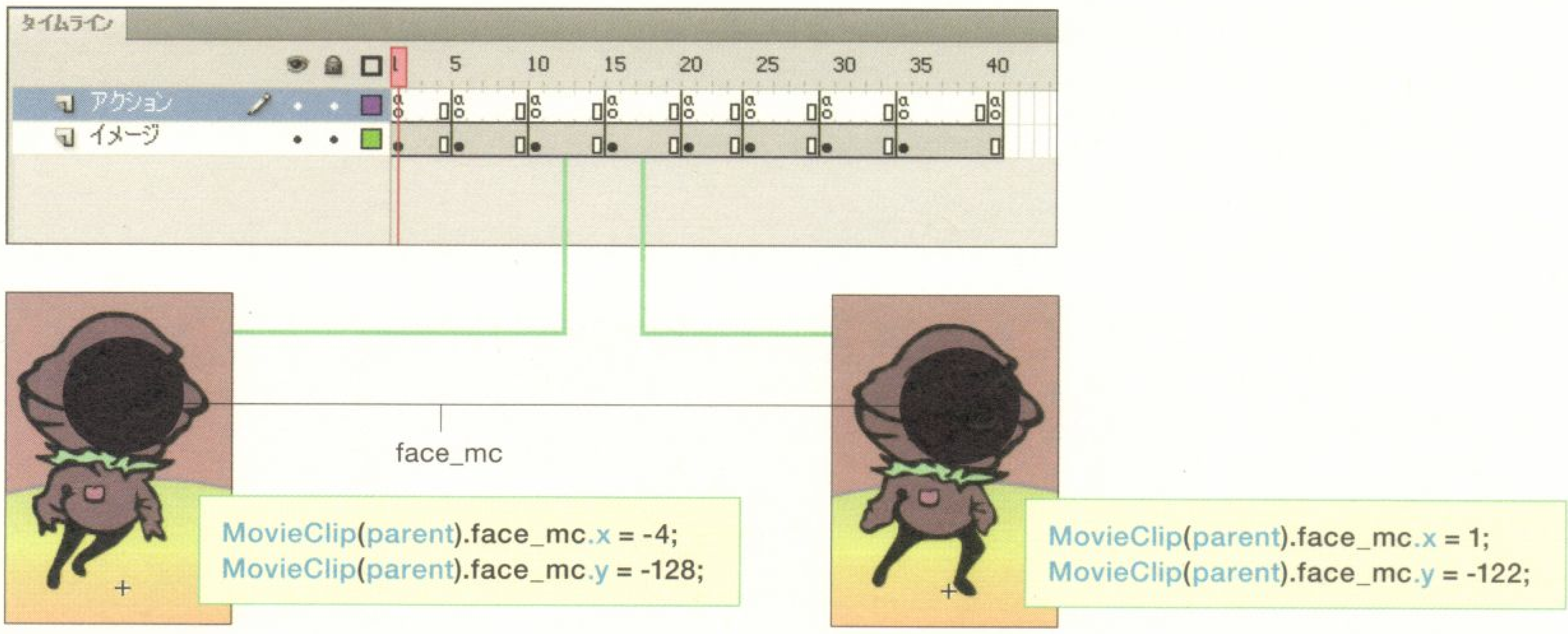
カメラに写っている映像を画像に変換して、画面上のキャラクターに貼り付けます。

1 顔写真を配置するためのムービークリップを「キャラクターMC」内に作成します。
このムービークリップはどのアニメーションでも使用するので、各アニメーションのムービークリップ内ではなく、「キャラクターMC」直下に置いています。また、作成したムービークリップは撮影した際と同様に円形のマスクをかけておきます。





2 アニメーション動作中に顔の位置を合わせるため、各アニメーションのムービークリップ内のタイムライン上に「顔MC」の位置を微調整するプログラムを書いておきます。



3 videoが表示しているカメラの映像を画像化するため、Bitmapインスタンスを作成し、Videoインスタンスが現在表示している画像を格納します。
作成したBitmapインスタンスは、「キャラクターMC」内に配置します。

メインタイムライン 1フレーム目

```
function onClickShutterBtn(event){
  camera_mc.shutter_btn.removeEventListener(MouseEvent.CLICK,onClickShutterBtn);

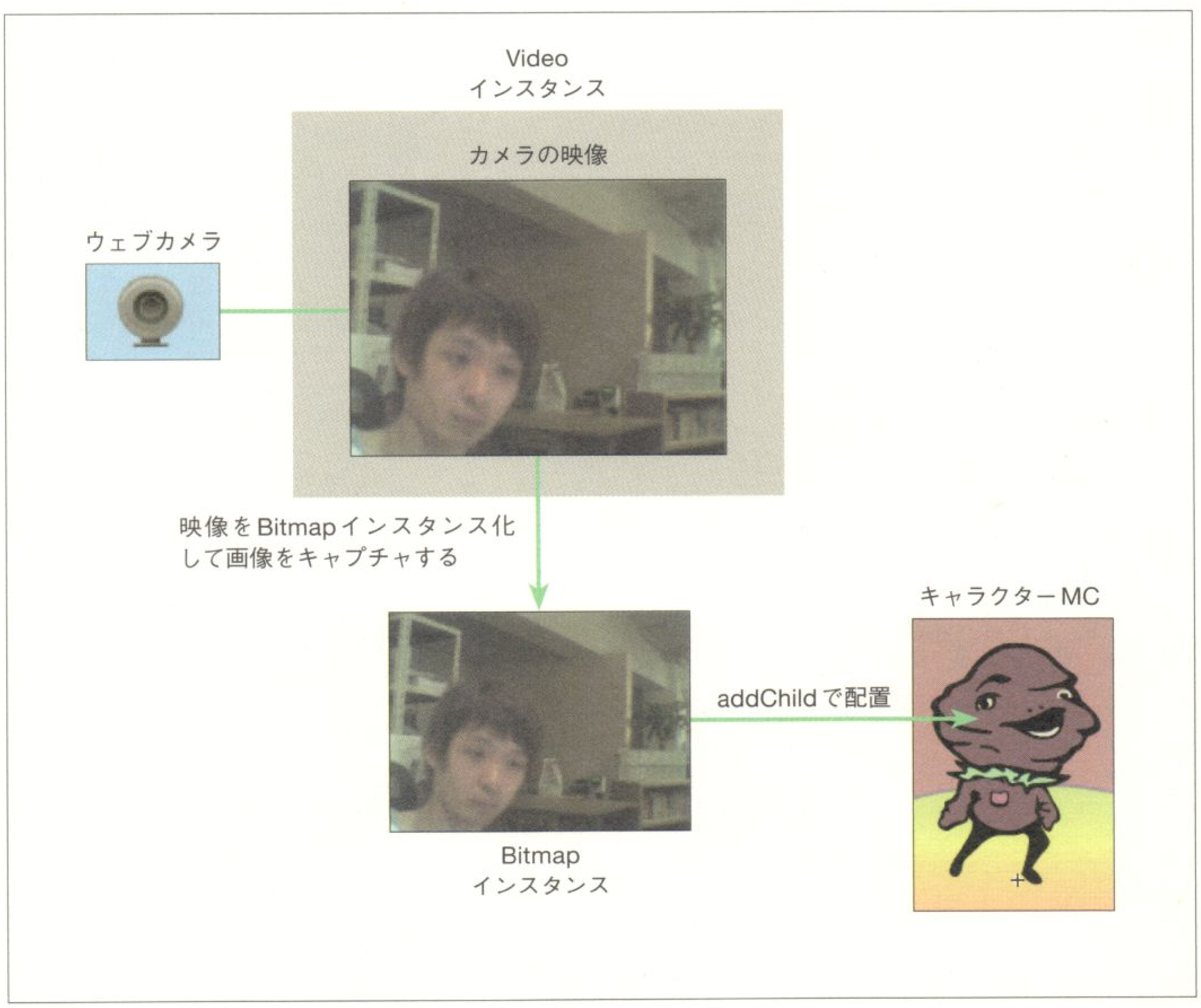
  //ビデオの映像を画像化して配置
  var bitmap_data = new BitmapData(video.width,video.height); ← ビデオのサイズのBitmapDataインスタンスを作成
  bitmap_data.draw(video); ← ビデオの映像を画像データとして格納
  var bitmap = new Bitmap(bitmap_data); ← 作成したBitmapDataインスタンスを持ったBitmapインスタンスを作成

  bitmap.scaleX = -1; ← 画像化したデータの位置を調整
  bitmap.x = bitmap.width/2; ← 画像化したデータの位置を調整
  bitmap.y = -bitmap.height/2; ← 画像化したデータの位置を調整

  character_mc.face_mc.face_frame.addChild(bitmap); ← キャラクターの顔に配置

  camera_mc.visible = false; ← 撮影に使ったムービークリップを非表示

  setKeyListeners(); ← キーの設定をするメソッドを実行
  playSound(); ← サウンドの再生をするメソッドを実行
}
```



STEP 4 キーボードでキャラクターを操作する

ユーザーによるキーボードの操作で、
キャラクターがアニメーションするようにプログラムします。

1 キーボードが押されたことを感知するために、KeyboardEvent.KEY_DOWNに対してリスナーを登録しています。尚、撮影後は常にキー入力に反応するようにしたいため、ステージに対してリスナーの登録を行っています。

シーン 1

メインタイムライン 1フレーム目

```
function setKeyListeners(){
  stage.addEventListener(KeyboardEvent.KEY_DOWN,onKeyboardDown);
}
```

■ キーの入力情報を取得する…………… P155へ

---

2 キーが押された際にどのキーが押されたのかを、KeyboardEvent.keyCodeを取得して判定します。押されたキーのキーコードが「スペースキー」「右矢印キー」「左矢印キー」「下矢印キー」のキーコードと等しい場合は、処理を実行します。
今回は、該当するボタンのムービークリップをアニメーションさせ、キャラクターのアニメーションを行うメソッド「playCharacter( )」に移動先のフレームラベルを引数として渡しています。

■ どのキーが押されたかを判定する…… P156へ

メインタイムライン 1フレーム目

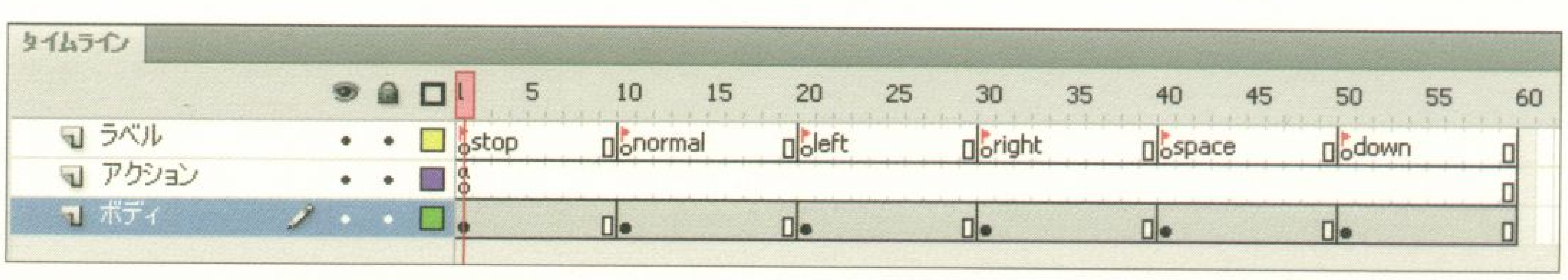
```
function onKeyboardDown(event){
  var key_code = event.keyCode;
  if(key_code == Keyboard.SPACE){
  ⬆ スペースキーが押された場合
    controler_mc.space_key.gotoAndPlay("start");
    playCharacter("space");
  }else if(key_code == Keyboard.DOWN){
  ⬆ 下矢印キーが押された場合
    controler_mc.down_key.gotoAndPlay("start");
    playCharacter("down");
  }else if(key_code == Keyboard.RIGHT){
  ⬆ 右矢印キーが押された場合
    controler_mc.right_key.gotoAndPlay("start");
    playCharacter("right");
  }else if(key_code == Keyboard.LEFT){
  ⬆ 左矢印キーが押された場合
    controler_mc.left_key.gotoAndPlay("start");
    playCharacter("left");
  }
}
```

---

3 メソッド「playCharacter( )」では、キャラクターのフレームを引数として渡されたフレームラベル名「target_frame」の位置に移動させます。

メインタイムライン 1フレーム目

```
function playCharacter(target_frame){
  character_mc.gotoAndStop(target_frame);
}
```

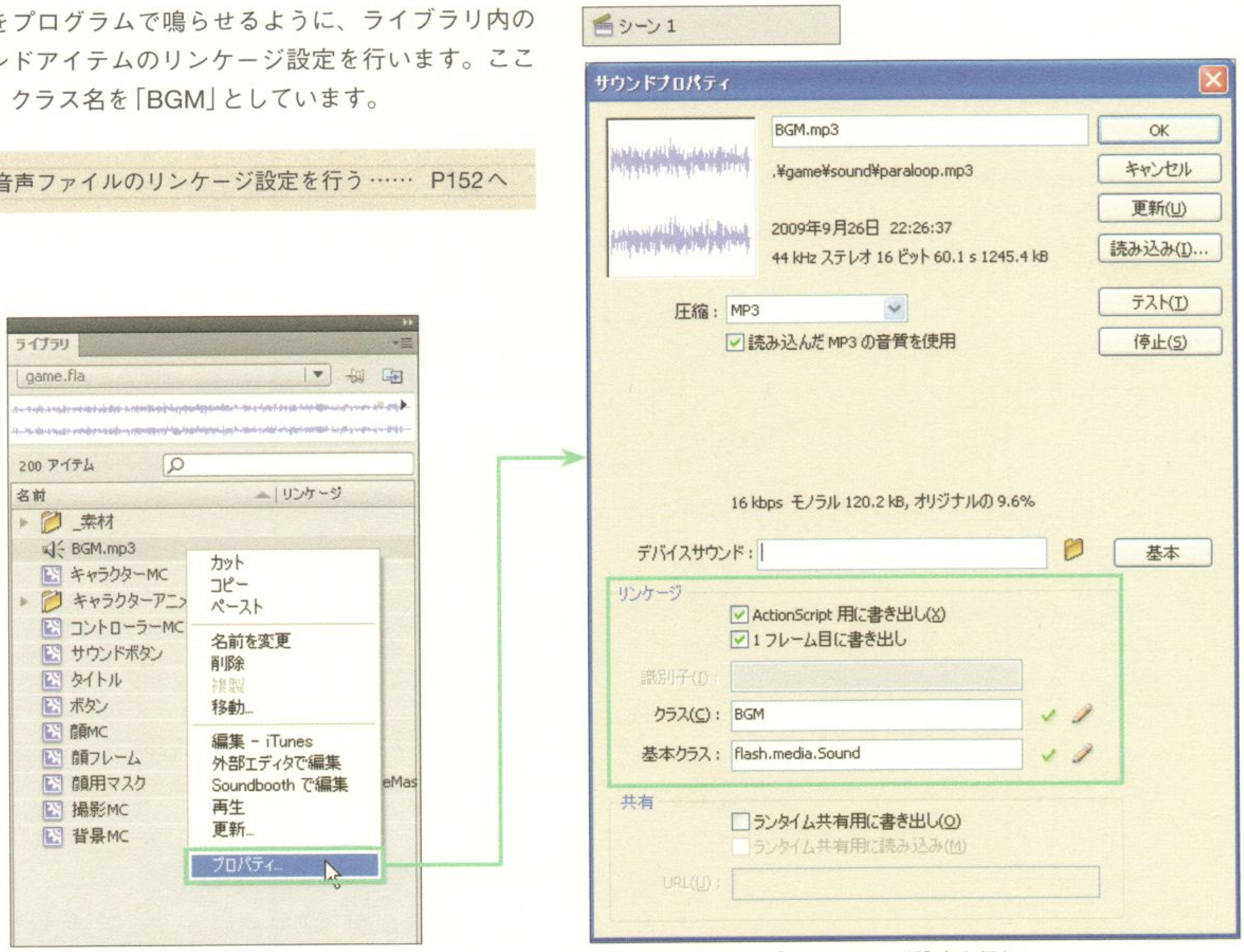


STEP 5 サウンドを制御する

ライブラリ内のサウンドを再生し、ボリュームのON/OFFができる仕組みを作ります。

1 音楽をプログラムで鳴らせるように、ライブラリ内のサウンドアイテムのリンケージ設定を行います。ここでは、クラス名を「BGM」としています。

■ 音声ファイルのリンケージ設定を行う …… P152へ



［サウンドプロパティ］でリンケージ設定を行う

2 音声のON/OFFを管理するための変数「sound_flg」と、再生する音声を制御するための変数「sound_channel」を宣言します。

メインタイムライン 1フレーム目

```
var sound_flg;
var sound_channel;
```

3 クラス名を「BGM」と設定したサウンドアイテムを呼び出し、「play( )」メソッドで再生を行います。また、再生時に取得できるSoundChannelインスタンスは、変数「sound_channel」に代入ておきます。

■ サウンドを動的に呼び出す……………………………………………………P153へ

メインタイムライン 1フレーム目

```
function playSound(){
  var bgm_sound = new BGM();
  sound_channel = bgm_sound.play();

  sound_flg = true; ⇐ sound_flgをtrueにして音声を再生の状態と分かるようにする

  controler_mc.sound_btn.buttonMode = true;
  controler_mc.sound_btn.addEventListener(MouseEvent.CLICK,onClickSoundBtn);
  ⇑ サウンドON/OFFのボタンにリスナーを登録
}
```

4 サウンドボタンクリック時に変数「sound_flg」の値を判定して、再生中の音声の音量を変更します。
「sound_flg」がtrueの場合は、音量が0に設定されたSoundTransformインスタンスを作成し、変数「sound_channel」のプロパティ「soundTransform」に代入します。「sound_flg」がfalseの場合は、音量をフルボリュームに設定したSoundTransformインスタンスを作成し、変数「sound_channel」のプロパティ「soundTransform」に代入します。

メインタイムライン 1フレーム目

```
function onClickSoundBtn(event){
  if(sound_flg == true){
    controler_mc.sound_btn.gotoAndStop("off");
    sound_flg = false;
    sound_channel.soundTransform = new SoundTransform(0,0);
  }else{
    controler_mc.sound_btn.gotoAndStop("on");
    sound_flg = true;
    sound_channel.soundTransform = new SoundTransform(1,0);
  }
}
```

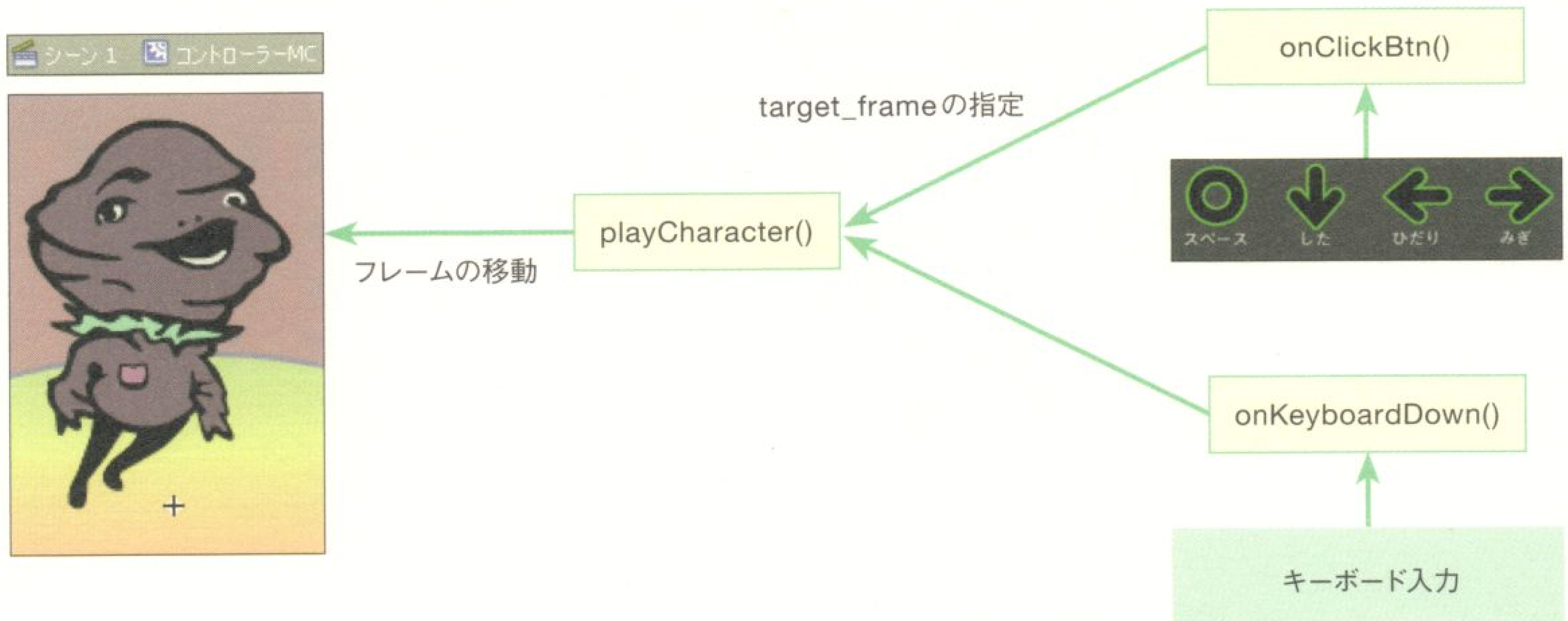


応用

Ex.1 クリックでもキャラクターを操作できるようにする

既に作成したメソッドを利用して、ユーザーのボタンクリックによる操作でも、
キャラクターをアニメーションさせられるようにします。

1 既に作成したメソッド「playCharacter( )」を利用することで、ボタンクリックとキーボード入力のどちらの操作でも、キャラクタをアニメーションさせられるようにします。
このように頻繁に使う処理をメソッドに分けておくことで、より効率的にプログラミングすることが可能です。



2 各キーのボタンそれぞれに対してボタンの設定を行います。
また、MouseEvent.CLICKのリスナーを登録します。

コントローラーMC 1フレーム目

```
space_key.buttonMode = true;
space_key.mouseChildren = false;
space_key.addEventListener(MouseEvent.CLICK,onClickBtn);
down_key.buttonMode = true;
down_key.mouseChildren = false;
down_key.addEventListener(MouseEvent.CLICK,onClickBtn);
left_key.buttonMode = true;
left_key.mouseChildren = false;
left_key.addEventListener(MouseEvent.CLICK,onClickBtn);
right_key.buttonMode = true;
right_key.mouseChildren = false;
right_key.addEventListener(MouseEvent.CLICK,onClickBtn);
```

3 eventのプロパティ「target」でクリックされた対象のMovieClipインスタンスが取得できるので、target_btnに格納してどのボタンと等しいかを判定しています。このようにif文を書くことで、1つのメソッドのみで複数のボタンの動作を書くことが可能です。
どのボタンか判定したのちは、メインタイムラインに作成したメソッド「playCharacter()」を該当のフレームラベルを引数として呼んでいます。

コントローラーMC 1フレーム目

```
function onClickBtn(event){
  var target_btn = event.target;
  target_btn.gotoAndPlay("start");
  if(target_btn == space_key){ ← スペースキーが押された場合
    MovieClip(parent).playCharacter("space");
  }else if(target_btn == down_key){ ← 下矢印キーが押された場合
    MovieClip(parent).playCharacter("down");
  }else if(target_btn == left_key){ ← 左矢印キーが押された場合
    MovieClip(parent).playCharacter("left");
  }else if(target_btn == right_key){ ← 右矢印キーが押された場合
    MovieClip(parent).playCharacter("right");
  }
}
```

PRACTICE 8

INDEX

INDEX

か行

さ行

た行

INDEX

著者プロフィール

株式会社クスール　http://www.cshool.jp/

クスールはFlashの技術を中心に、“ものづくり”を教える小さな学校です。授業は作品づくりを中心としたカリキュラムで、受講者自らが手を動かすことで技術を身につけることを目的としています。また、講座以外にもイベントの開催、書籍の執筆を行うことで、クリエイター交流の場や、学ぶ機会を提供しています。コースは1年を通じて開催していますので、興味のある方は気軽に見学に来てください。

■松村 慎（まつむら しん）

2002年、カナダ・バンクーバーから帰国したのち、株式会社バスキュールにてFlashデベロッパーとして勤務。2004年にFlashのActionScriptを中心に、ものづくりを教える教室クスールを開設。2006年春からは京都精華大学デザイン学部ビジュアルデザイン学科にて、非常勤講師としても勤務。得意技はビールのあとの焼酎ロック(芋)。

■加賀谷 和（かがや のどか）

WEBデザイナー。1980年ののどかな日に生まれる。多摩美術大学卒。数社を経て、現在、株式会社クスール所属。肩書きは担任。プロモーション系ウェブサイトを中心に、デザイン、ディレクションに従事。好物はクロワッサン。路地裏と高いところからの眺望が好き。得意技は迷子になること。

■尾崎 俊介（おざき しゅんすけ）

株式会社クスール クリエイティブディレクター兼教頭。Webのキャンペーンサイトを中心に、ディレクションや制作を手がける。その一方で、「自社のFlash講座」や「京都精華大学の非常勤講師」など講師としても活動。また、電子工作などを取り入れた、研究開発チーム「くるくる研究室」に所属。得意技は、出席の確認と指し棒で、壁を叩くこと。

■大西 拓人（おおにし たくと）

株式会社 クスール Flashエンジニア。1985生まれ。早稲田大学第一文学部文芸専修卒。大学時代に早稲田大学広告研究会にて広告について学ぶ。広告サイト制作やアプリケーションの開発などをする傍ら、ActionScriptの講座「ほんきでクスール」にてFlashでのものづくりを教えている。得意技は寝返りを打つこと。受賞暦：2009年度 グッドデザイン賞。Yahoo Internet Creative Award 2008銅賞。個人ブログ：http://aretokore.jp/blog/

■上泉 洋介（かみいずみ ようすけ）

株式会社 クスール主任。大学卒業後、カナダ・バンクーバーでWebの技術と嫁を手に入れ、2001年に帰国。事業会社で7年間Webマスターとして勤務したあと、2008年にクスールに入社。現在は学校事業部マネージャとして講座全体の運営を行う傍ら、講師としても活動中。ときどきふらりと釣りに行きます。得意技はおにぎりの買い食い。

■岡崎 大典（おかざき だいすけ）

フリーランスのFlash・Silverlightなどリッチクライアントディベロッパー／サーバサイドエンジニア／テクニカルディレクター。2002年にウェブ業界への就職を機に地元北海道から上京。WebエンジニアやFlashエンジニアとして、都内の制作会社に勤務。2006年にフリーランスとしてdynalogueという屋号で活動を開始。現在は設計・制作・コンサルティングなどを中心に活動している他、講師や講演も行っている。趣味は3匹の愛犬と遊ぶことと、ドライブと、スポーツ鑑賞(野球、サッカー、バスケ、アメフト、格闘技、等々幅広く)。得意技はしれっとラーメンを早食いすること。クスールではもっとほんきでクスールの講師を担当。

STAFF

PART3制作・執筆担当

PRACTICE 1　執筆・サンプルデザイン：加賀谷和、キャラクターデザイン：中里寛
PRACTICE 2　執筆：上泉洋介、サンプルデザイン：加賀谷和
PRACTICE 3　執筆：尾崎俊介、サンプル提供：倉片友子
PRACTICE 4　執筆：尾崎俊介、サンプル提供：倉片友子
PRACTICE 5　執筆：松村慎、サンプルデザイン：加賀谷和、写真提供：足成（http://www.ashinari.com/）
PRACTICE 6　執筆：岡崎大典、サンプル提供：堀内香織
PRACTICE 7　執筆：大西拓人、サンプルデザイン：加賀谷和、撮影協力店：BAR TRAM、写真提供：TAKU
PRACTICE 8　執筆：大西拓人、サンプルデザイン：大賀光洋（http://www.grapinco.jp/）

ブックデザイン：三宮暁子（アレフ・ゼロ）
カバー・本文イラスト：2g（http://twograms.jimdo.com/）
DTP：ayumu print
編集：角竹輝紀

Flash(フラッシュ)クリエイターになるための教科書(キョウカショ)

2010年1月29日　初版第1刷発行

著者　クスール
発行者　中川信行
発行所　株式会社 毎日コミュニケーションズ
〒100-0003　東京都千代田区一ツ橋1-1-1　パレスサイドビル
TEL：048-485-2383（注文専用ダイヤル）
TEL：03-6267-4477（販売）
TEL：03-6267-4431（編集）
E-mail：pc-books@mycom.co.jp
URL：http://book.mycom.co.jp
印刷・製本　株式会社 ルナテック

ISBN978-4-8399-3165-0

関連書籍

基本からしっかりわかる ActionScript 3.0

森巧尚［著］

ISBN978-4-8399-3025-7

ActionScript 3.0 を基本からきっちりマスターしたい人のための本。
図解とイラストを豊富に用いて、やさしくわかりやすく解説。

Flash Lite で作る携帯コンテンツ実践教科書

東條雅樹・岡田昇三・よしだゆたか［著］

ISBN978-4-8399-3024-0

Flash Lite を使った携帯 Flash コンテンツの作成テクニックの解説書。
待ち受けスライドショー、時計、おみくじ、モグラたたきゲーム、
神経衰弱ゲームなど、実際の作例を制作しながら、
携帯 Flash 制作のコツとテクニックをマスターしよう。

ActionScript 3.0 プロフェッショナルガイド

野中文雄［著］

ISBN978-4-8399-2652-6

入門者から取り組める、最強の ActionScript 3.0 解説書。
ActionScript 3.0 を徹底的にマスターしたい人のための手引き書です。
サンプルの制作を通して、スクリプティングの
「基本」「考え方」「具体的な組み立て方」までが体得できます。

ISBN978-4-8399-3165-0
C3055 ¥2480E

9784839931650

定価：本体2,480円＋税

1923055024809

Flashクリエイターになるための教科書



クスール［著］

マイコミ